Overview of BOUT++

Ben Dudson, D.Dickinson, L.Easy, P.Hill, J.Leddy, S.Mekkaoui, B.Shanahan, and the BOUT++ team

York Plasma Institute, Department of Physics,
University of York, Heslington, York YO10 5DD, UK

BOUT++ Workshop, LLNL

16th December 2015

What is BOUT++?

- A toolbox for solving PDEs on parallel computers. Aims to reduce duplication of effort, and allow quick development and testing of new models
- A collection of examples and test cases
- Focussed on flute-reduced plasma models in field-aligned coordinate systems, though has more general capabilities

And what is it not:

- A single plasma model or simulation
- A general library of numerical methods. Other libraries like PETSc are available for that
- Magic. Appropriate numerical methods depend on the problem, and must be chosen intelligently by the user

BOUT++: A toolbox for plasma simulations

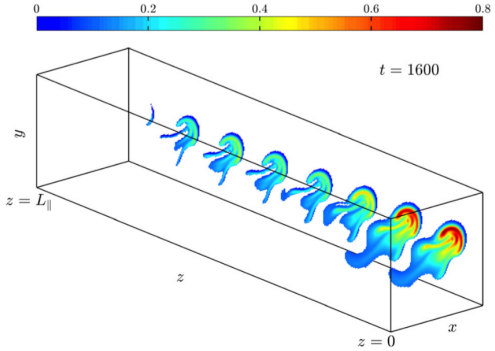
- Collection of useful data types and associated routines. Occupies a middle ground between problem-specific codes and general libraries (e.g. PETSc, Trilinos, Overture, Chombo,...)
- Has its origins in the BOUT code[1][2][3]. Re-written and re-designed (at least once) in C++[4][5]
- Researchers can make use of a (mostly) well tested library of simulation code and input / output tools
- Greatly reduces the time needed to develop a new simulation

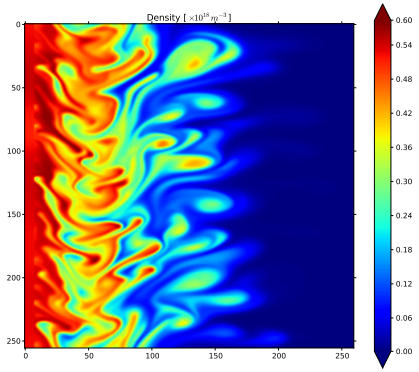
---

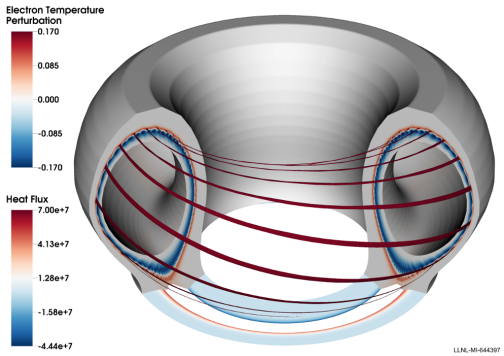
[1] X.Q. Xu and R.H. Cohen, Contrib. Plasma Phys. 38, 158 (1998)
[2] Xu, Umansky, Dudson, Snyder, CiCP 4, 949-979 (2008)
[3] Umansky, Xu, Dudson et al. Comp. Phys. Comm. 180, 887-903 (2008)
[4] Dudson, Umansky, Xu et al. Comp. Phys. Comm 180, 1467 (2009)
[5] Dudson et al. J. Plasma Phys. (2014).
Preprint: `http://arxiv.org/abs/1405.7905`

Applications in tokamak physics

- Filaments / blobs
  - Transport of heat and particles in SOL
  - 2D examples run on a laptop
- Edge turbulence
  - Formation of blobs
  - Near SOL heat transport (λ_q)
- Divertor simulations
  - Spreading of particle and power fluxes to surfaces
  - Interaction with neutral gas and detachment
- Pedestal physics and ELMs
  - Gyro-fluid models to capture FLR, Landau damping, and drift resonance effects
  - L-H transition physics
  - Stability and nonlinear dynamics

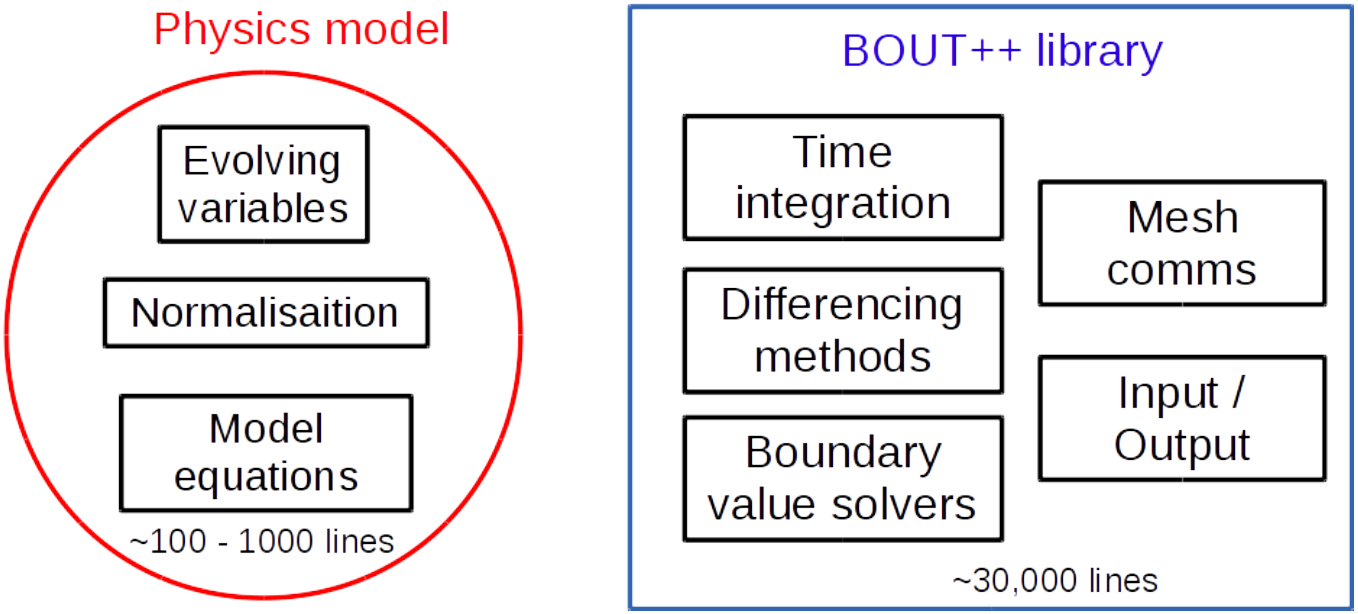






Getting started: BOUT++ code structure

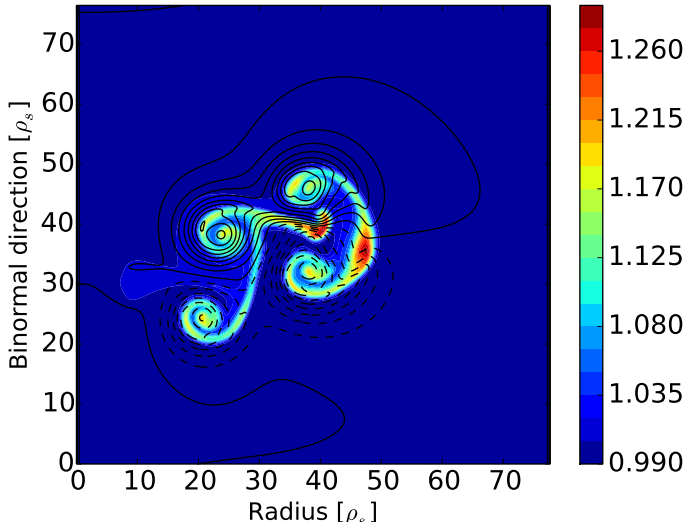
- Separates generic methods from model-specific code
- Most of the code doesn't know or care about what a variable represents, its normalisation etc. Only needs to know the geometry and which operation to perform

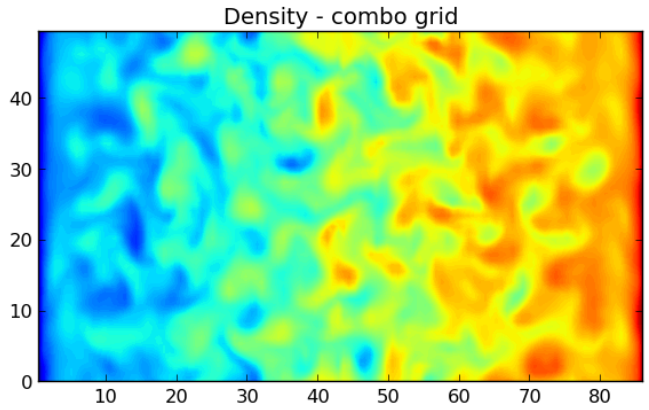


Getting started: Examples

Many examples and test cases are available in the **examples/** subdirectory

- **conduction/** : Temperature conduction
- **blob2d/** : 2D blob propagation
- **hasegawa-wakatani** : 2D drift-wave turbulence
- **sod-shock/** : Standard 1D fluid shock problem
- **orszag-tang/** : 2D MHD problem
- **lapd-drift/** : Turbulence in LAPD linear device
- **elm-pb/** : ELM simulations





Example : `examples/conduction`

This solves heat conduction in 1D (in y):

$$\frac{\partial T}{\partial t} = \nabla \cdot (\chi \partial_{\parallel} T)$$

Two variables are needed: T and χ (chi). In the code we define

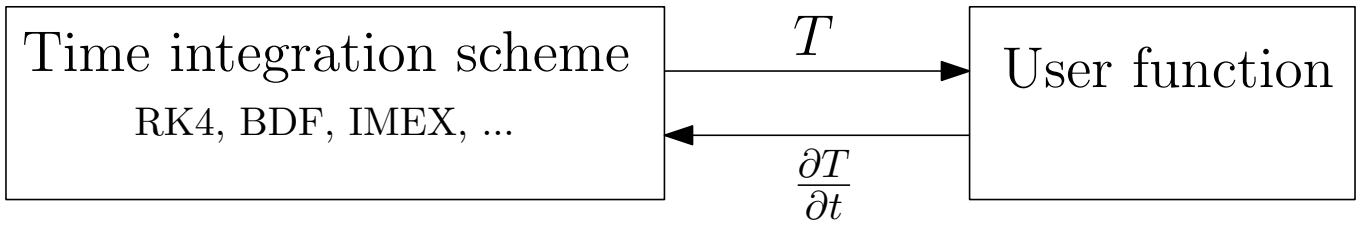
```
Field3D T;
BoutReal chi;
```

- BoutReal is just an alias for double
- Field3D is a BOUT++ **class** or type, which handles 3D arrays.
  [Defined in `include/field3d.hxx`, code in `src/field/field3d.cxx`]

Time evolving variables

- Time and space are discretised separately: Method of Lines
- Time integrators solve a generic ODE of the form

$$\frac{\partial \underline{f}}{\partial t} = F\left(\underline{f}, t\right)$$



To implement a model, we need to:

1. Specify the variables to evolve $\underline{f}$ (here only T)
2. Implement the "RHS" function $F\left(\underline{f}, t\right)$

Physics model parts

Every physics model has two parts:

1. An initialisation function which is called (run) once at the start of a simulation
2. A run function which is usually called every time step

In the **`examples/conduction`** code, these appear as two C-style functions

```
int physics_init(bool restarting) {

  return 0;
}

int physics_run(BoutReal t) {

  return 0;
}
```

Physics model parts

Every physics model therefore has two parts:

1. An initialisation function which is called (run) once at the start of a simulation
2. A run function which is usually called every time step

For those who prefer a more C++ style interface, **`examples/conduction-newapi`**:

```
class Conduction : public PhysicsModel {
  protected:

  int init(bool restarting) {
    return 0;
  }

  int rhs(BoutReal t) {
    return 0;
  }
};
```

Numerical methods - Time integration

- Method of Lines: fully explicit, fully implicit or IMEX
- Implicit time integration methods operate Jacobian-free
- Users only need to implement the (nonlinear) function $F(\cdot)$ which operates on state vector $\mathbf{f}$:

$$\frac{\partial \mathbf{f}}{\partial t} = F(\mathbf{f})$$

- Standard explicit methods e.g. RK4, Karniadakis with or without adaptive timestepping
- Implicit methods through the PETSc and SUNDIALS libraries $\rightarrow$ **Most simulations use:** adapive order, adaptive timestep BDF method from SUNDIALS
- Optional preconditioning also implemented. Problem specific, usually "physics-based" preconditioner

Numerical methods - Spatial operators

A number of operators are available, which users combine to implement models e.g

$$\frac{\partial n}{\partial t} = -[\phi, n] + 2\frac{\rho_s}{R_c}\frac{\partial n}{\partial z} + D_n \nabla_\perp^2 n$$

becomes:

```
ddt(n) = - bracket(phi, n, BRACKET_ARAKAWA)
         + 2*DDZ(n)*(rho_s/R_c)
         + D_n*Delp2(n)
```

- The method to be used can be set globally (input file or command-line), per-dimension, or per-operator
- Diffusion-like operators: 2^{nd}-order, 4^{th}-order central difference or 3^{rd}-order CWENO
- Advection operators: Arakawa, CTU, 1^{st}-order and 4^{th}-order upwind, 3^{rd}-order WENO

BOUT++ components

- The majority of the BOUT++ code is a library of finite difference routines for operating on `Field` objects (arrays)
- Each component has one interface and (usually) many implementations
  - Time integration (CVODE/RK4/IMEX/SLEPc/...)
  - File reading and writing (NetCDF/HDF5/PDB)
  - Laplacian inversion (Tridiagonal/PETSc/...)
  - Boundary conditions (Neumann/Dirichlet/Robin/...)
- You don't have to use any of these routines; the only thing you need is a Mesh

Factory pattern

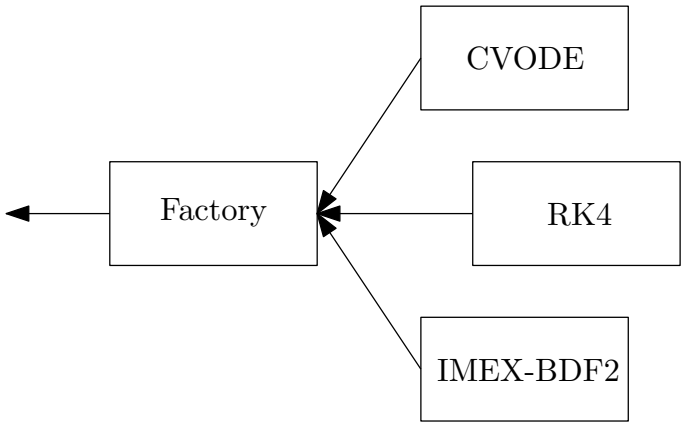
An isolated part of the code (the “factory”) is the only part which “knows” about the different implementations.

When you need to create a solver :

```
solver = Solver::create();
```

The solver factory then:

1. Reads the “type” option
2. Creates the chosen solver
3. Returns the same thing (a `Solver*`) regardless of implementation



The main benefits are:

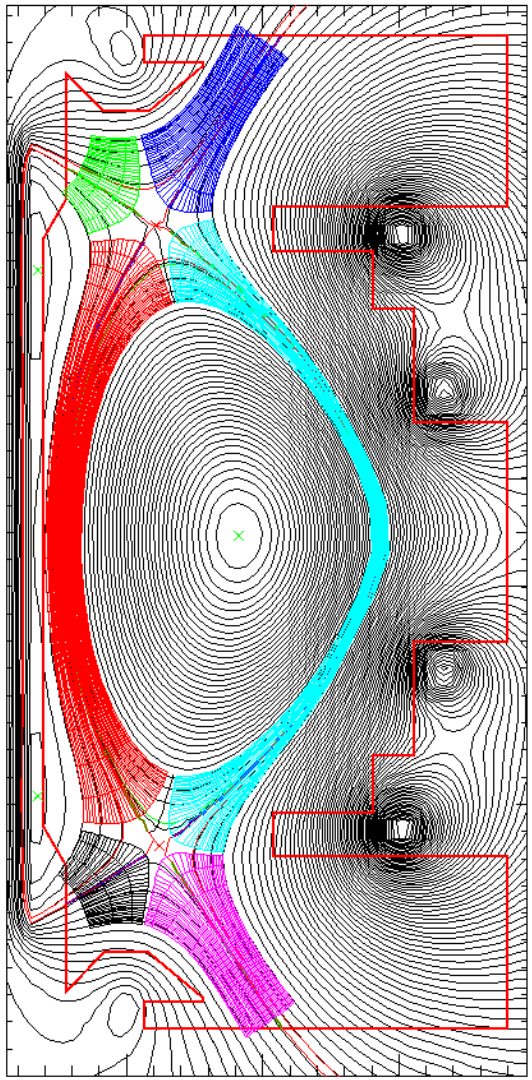
- Different components can be used without recompiling
- New implementations added without changing user code

Pre- and Post-processing

See **`tools/`** sub-directory

- Input to BOUT++ can come from
  - Analytic expressions
  - Simplified grid generators e.g. flux tube (Cyclone), theta pinch, shifted circle tokamak
  - TEQ (DSKGATO), ELITE formats
  - EFIT equilibria (g-file)
  - VMEC ... coming soon!

⇒ More discussion tomorrow on meshes

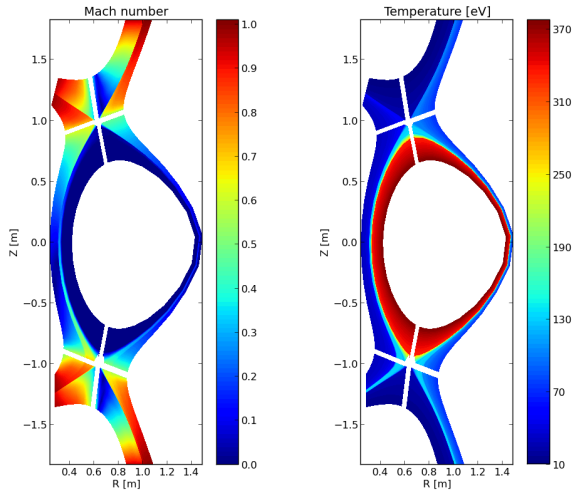


Hypnotoad grid generator

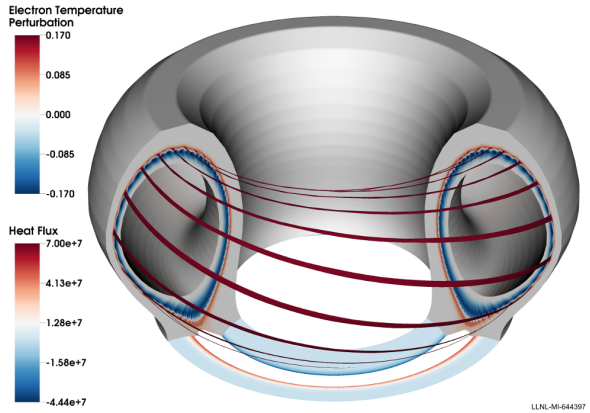
Pre- and Post-processing

See **`tools/`** sub-directory

- Input to BOUT++ can come from
  - Analytic expressions
  - Simplified grid generators e.g. flux tube (Cyclone), theta pinch, shifted circle tokamak
  - TEQ (DSKGATO), ELITE formats
  - EFIT equilibria (g-file)
  - VMEC ... coming soon!
- Output to binary files (NetCDF/PDB/HDF5)
  - Routines to read into Python, IDL, Matlab, Mathematica, Octave
  - Interfaces to Mayavi and VisIT for 3D rendering



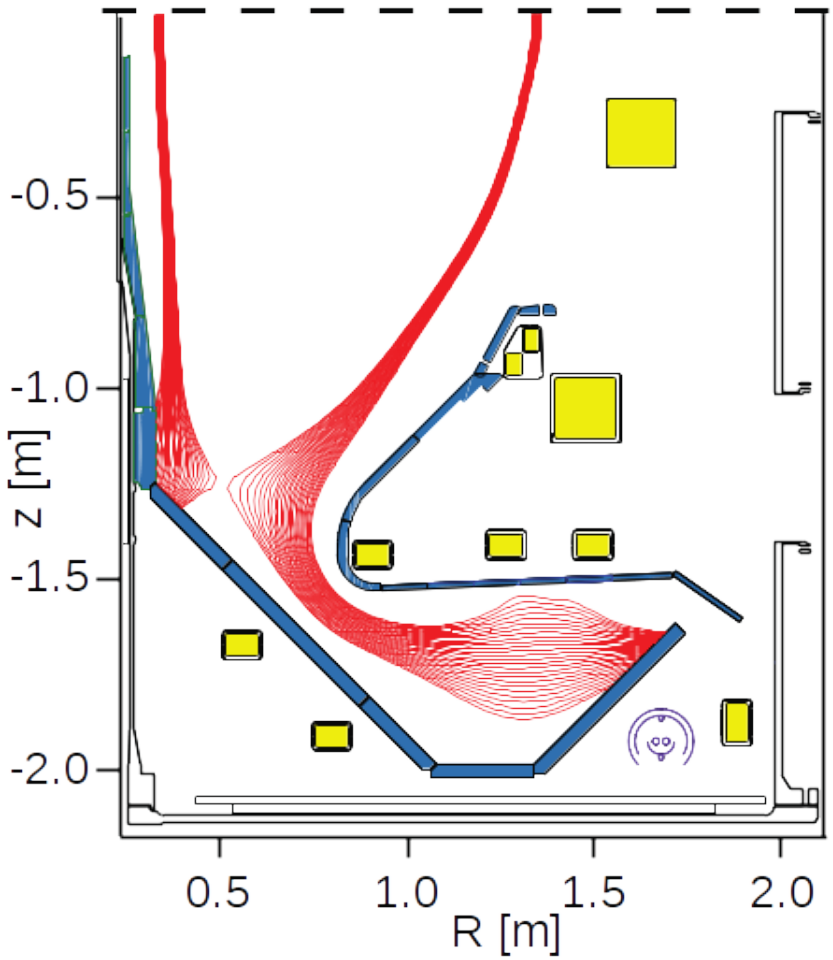
Transport in MAST



ELM in DIII-D

Development driven by physics needs

A major focus in the UK is edge and divertor physics



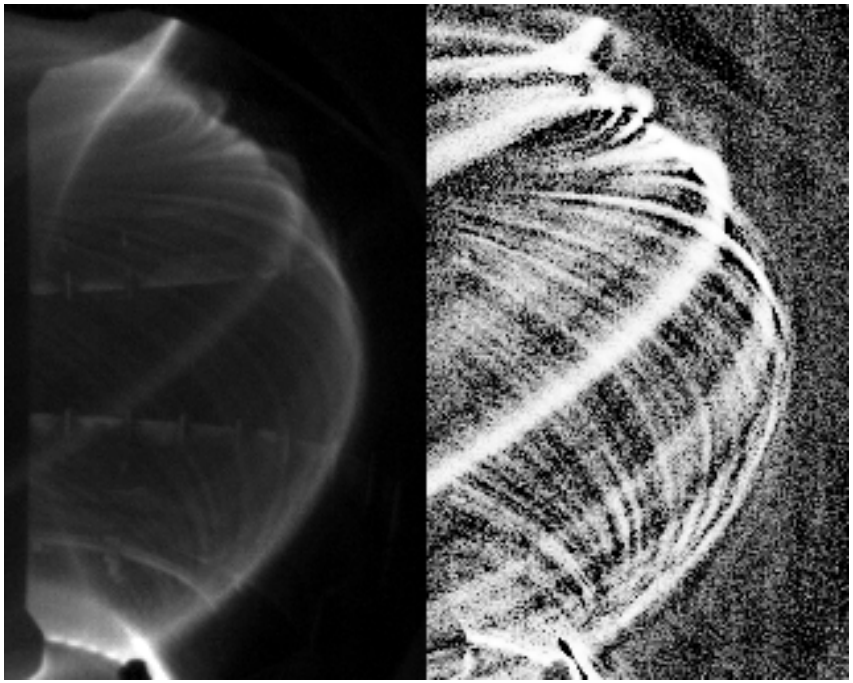
MAST-Upgrade (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics



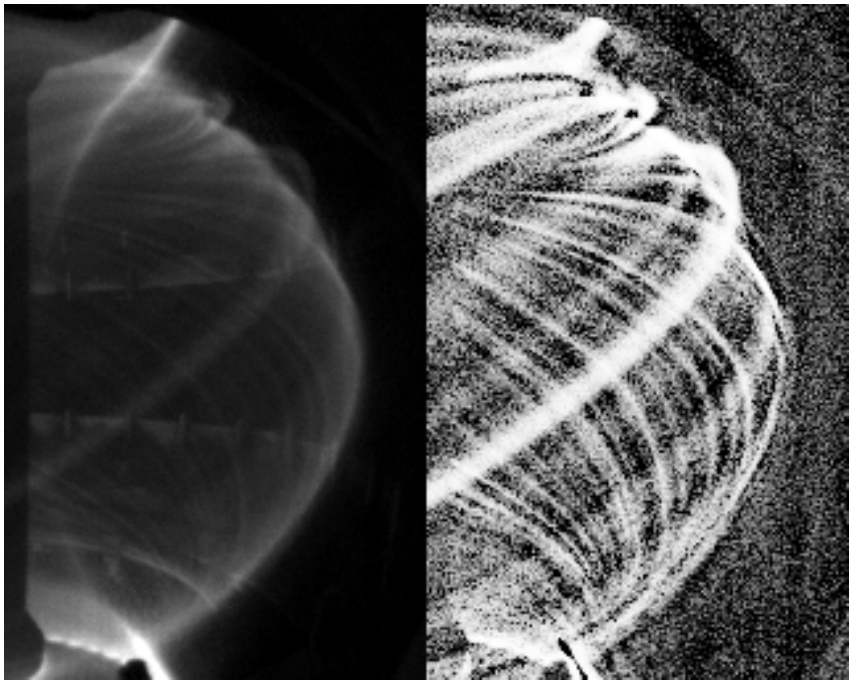
N.Walkden (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics



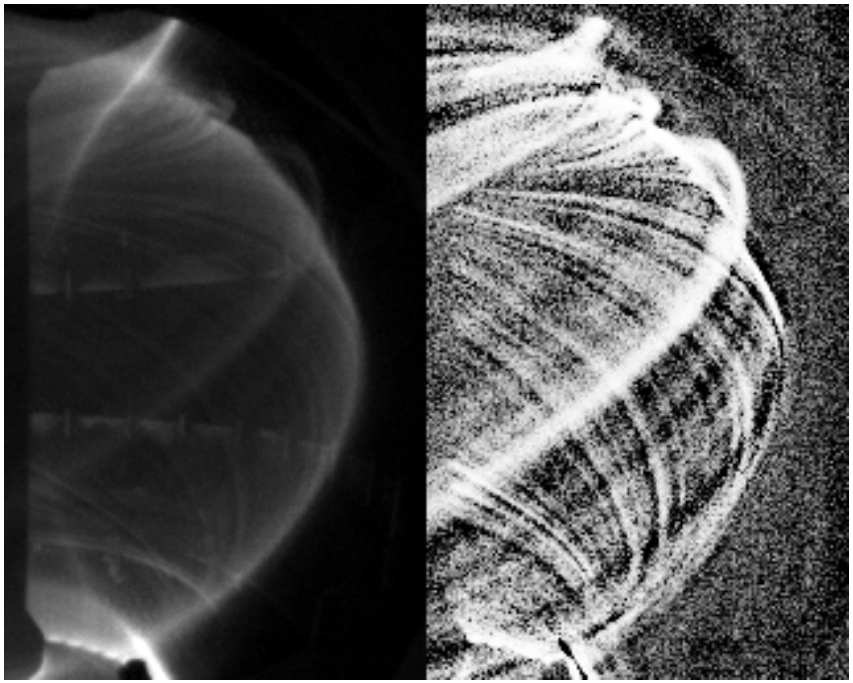
N.Walkden (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics



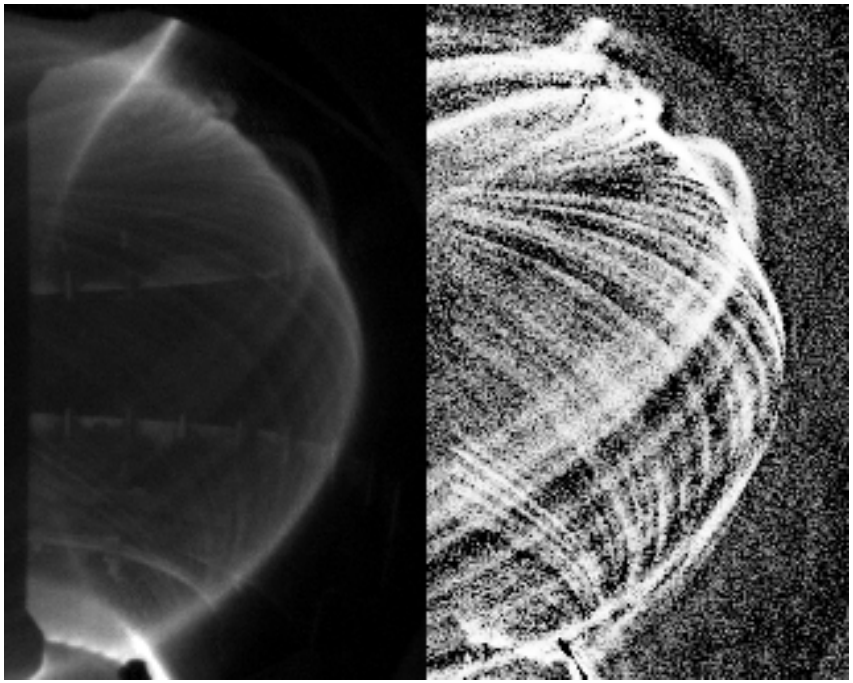
N.Walkden (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics



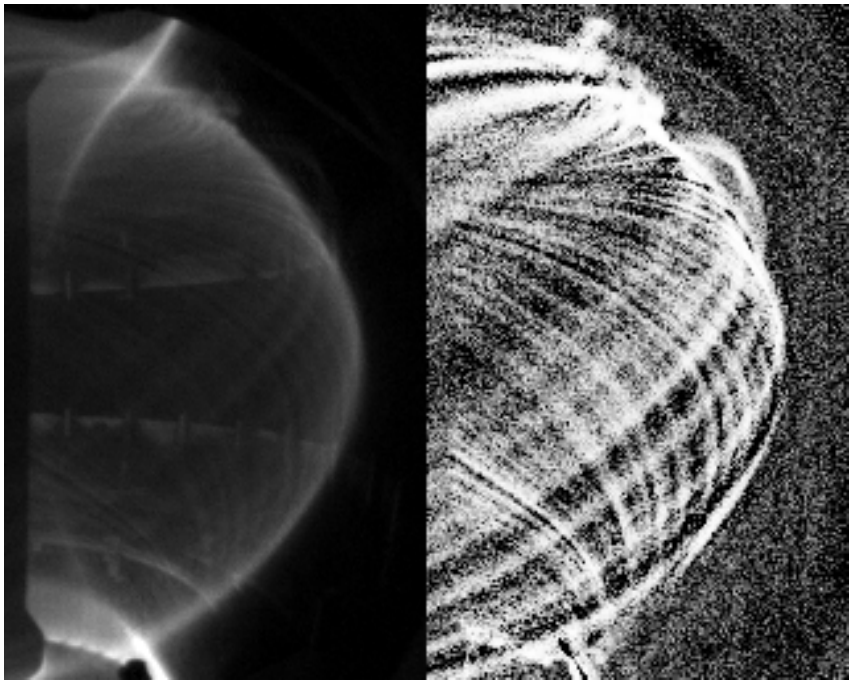
N.Walkden (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics



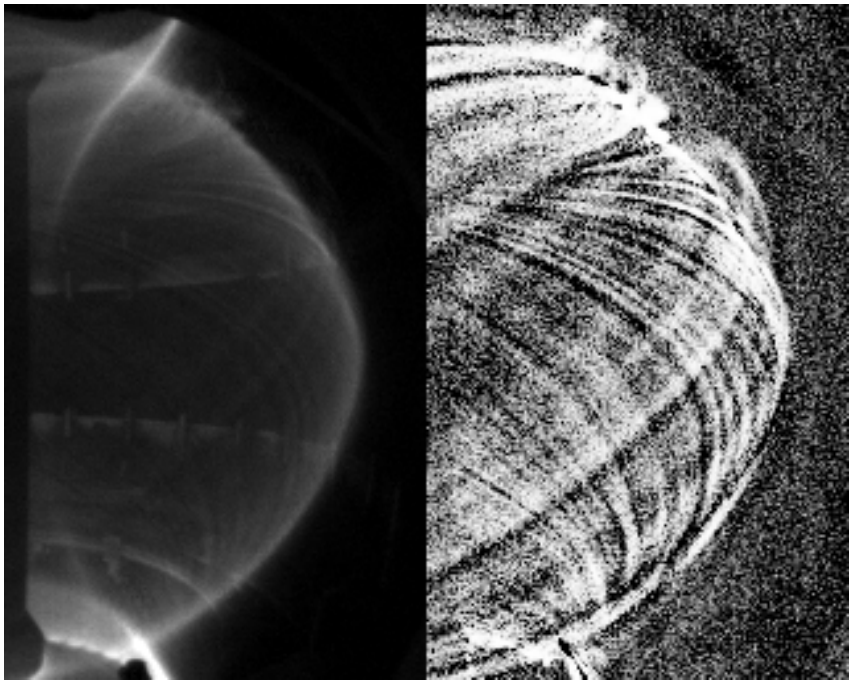
N.Walkden (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics



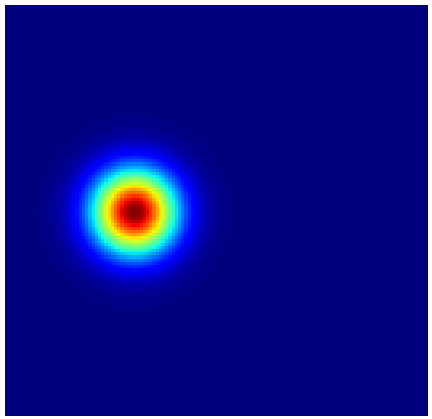
N.Walkden (CCFE)

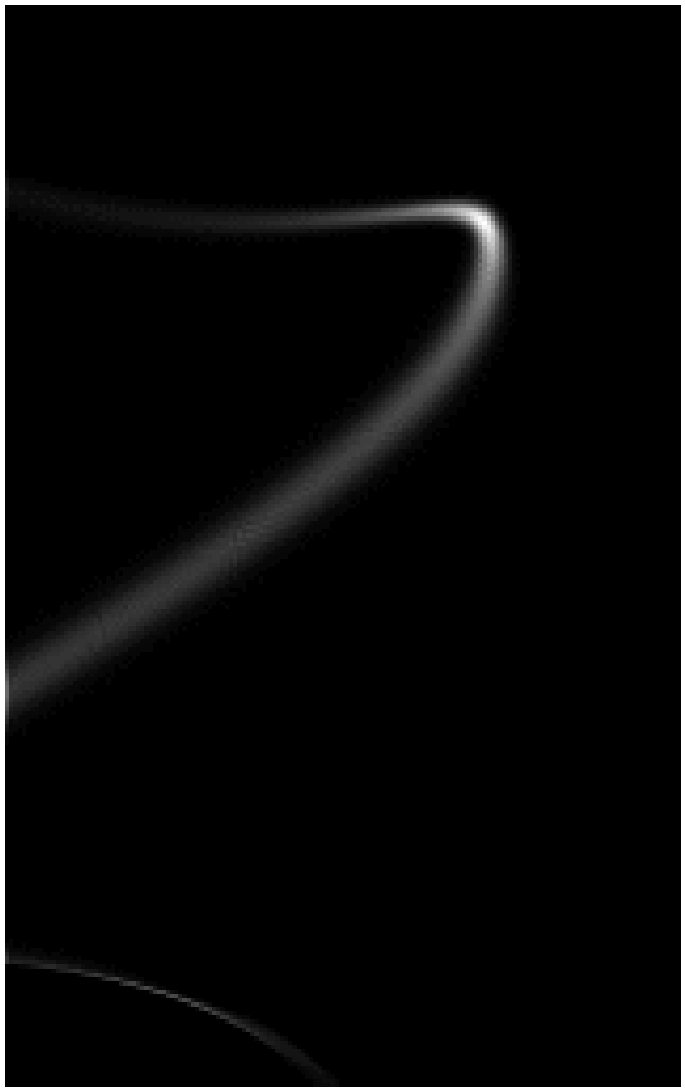
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





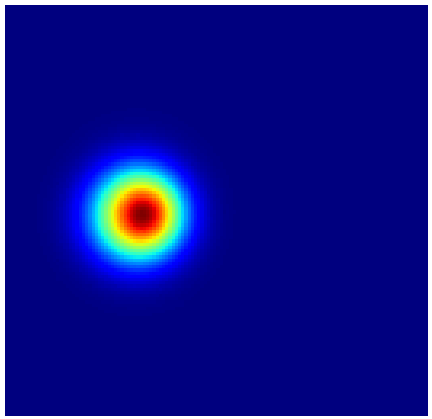
N.Walkden (CCFE)

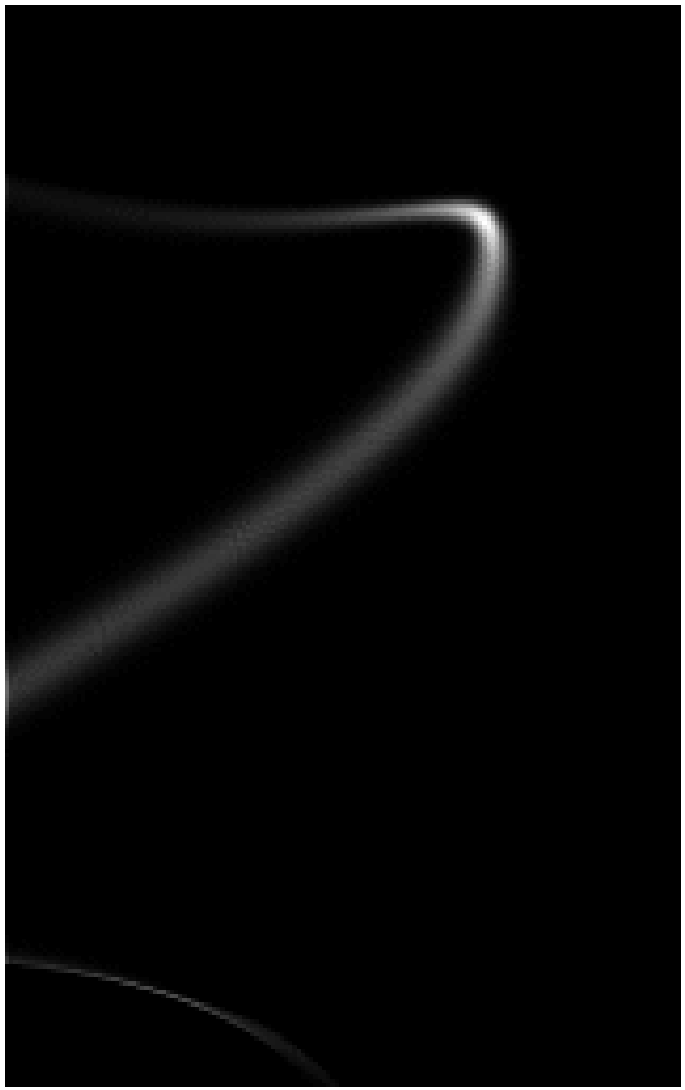
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





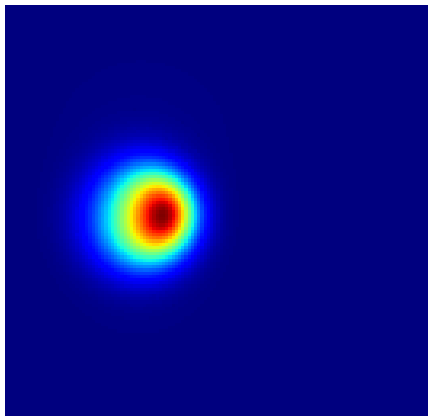
N.Walkden (CCFE)

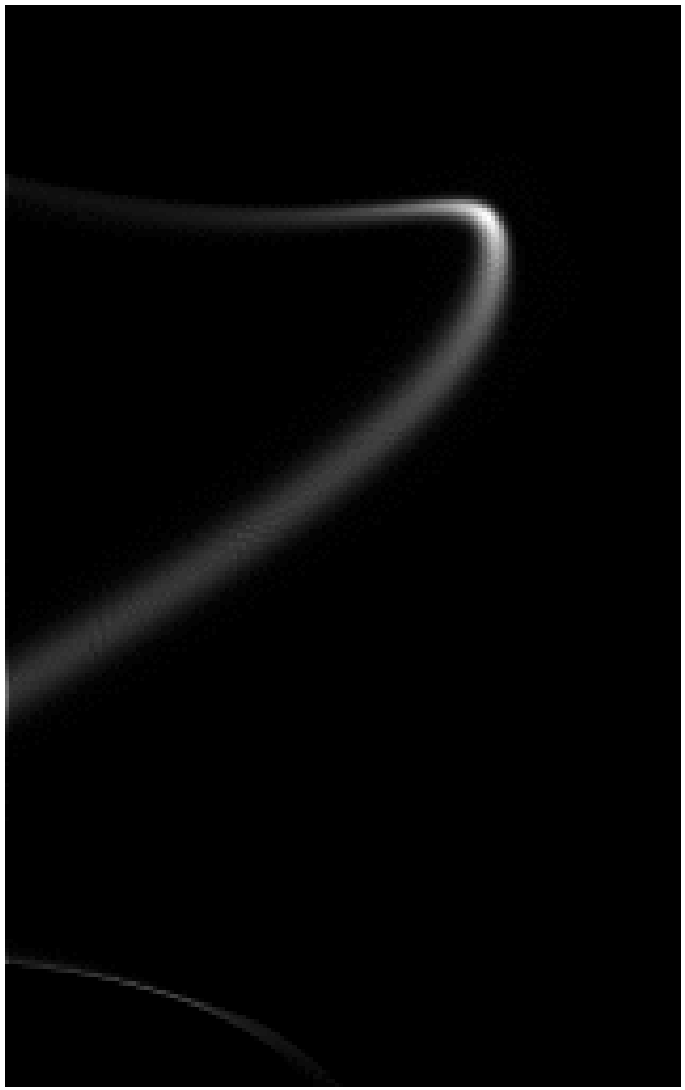
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





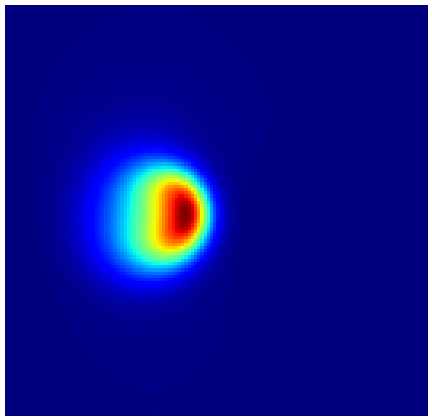
N.Walkden (CCFE)

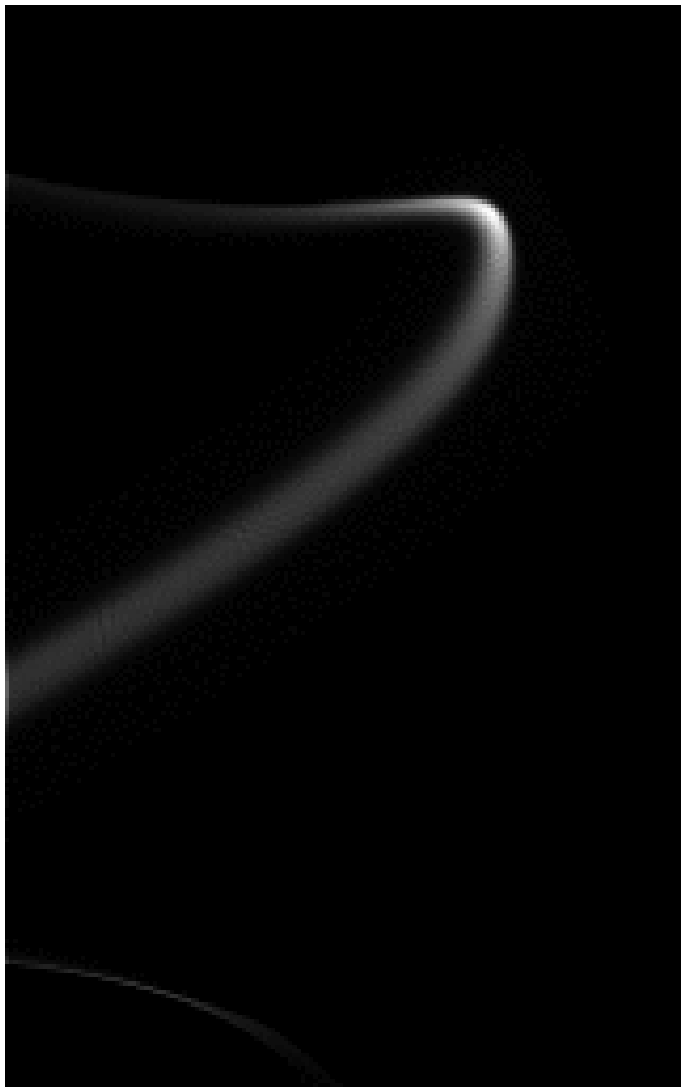
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





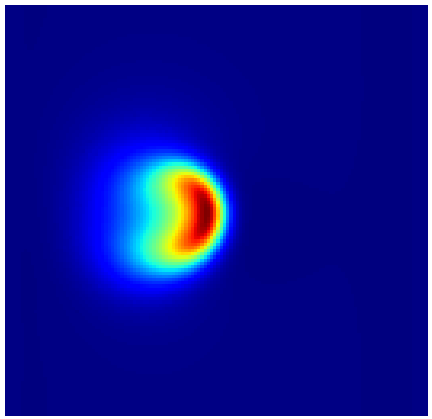
N.Walkden (CCFE)

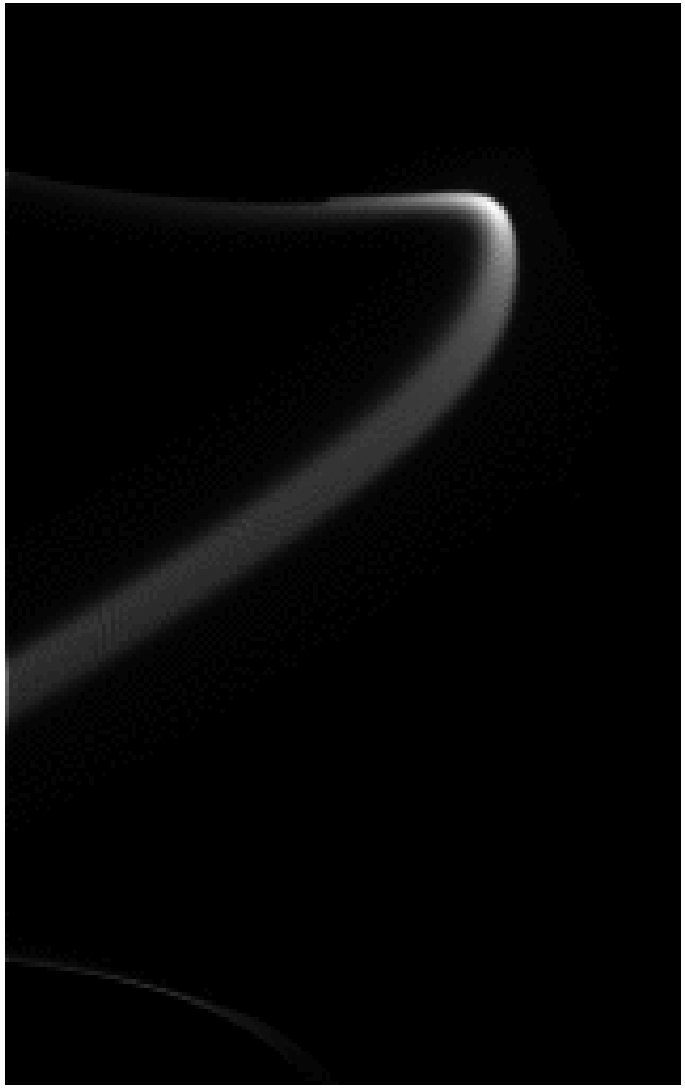
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





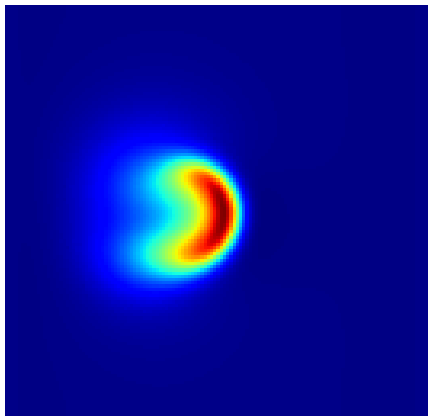
N.Walkden (CCFE)

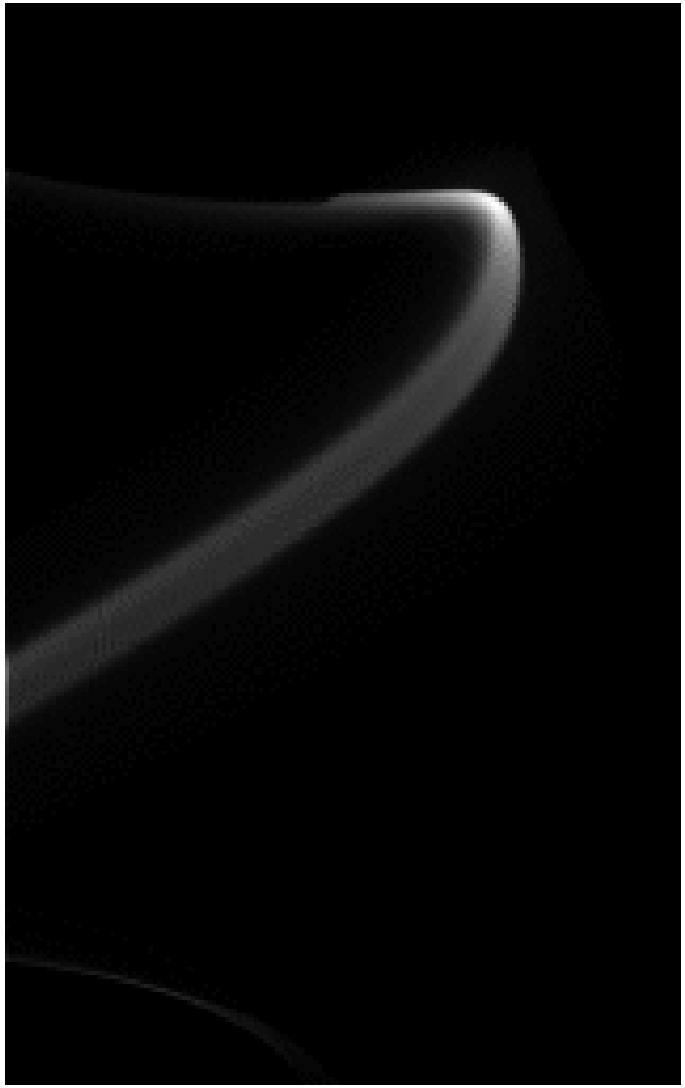
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





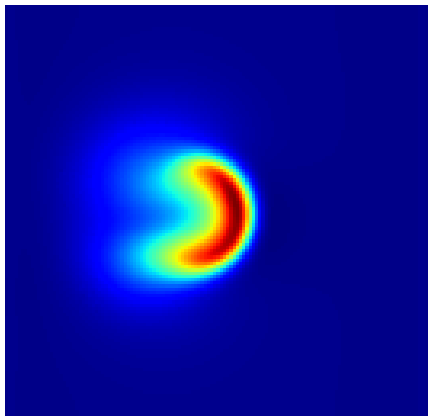
N.Walkden (CCFE)

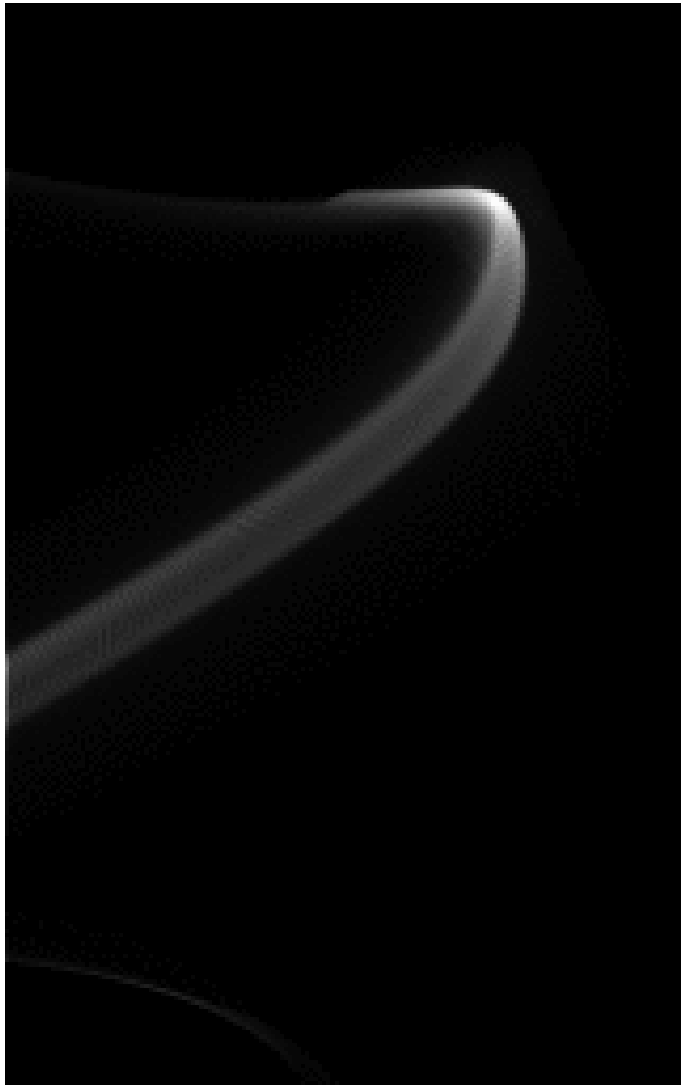
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





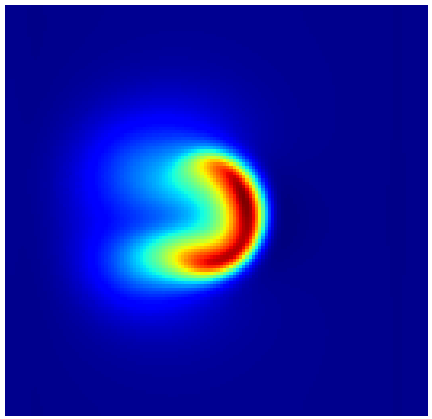
N.Walkden (CCFE)

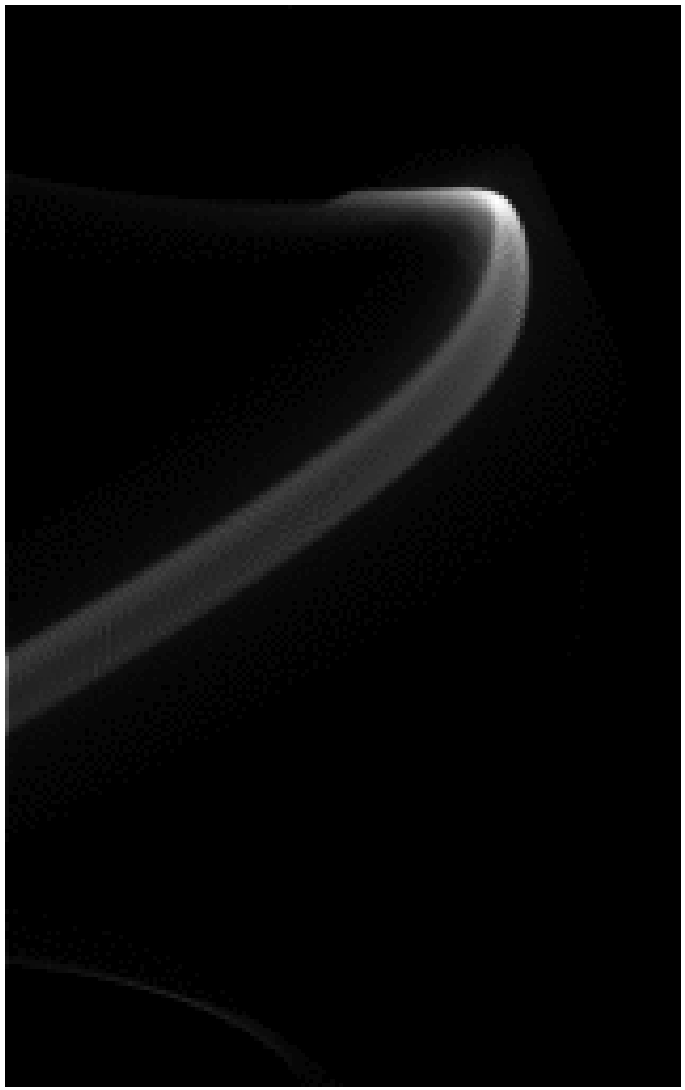
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





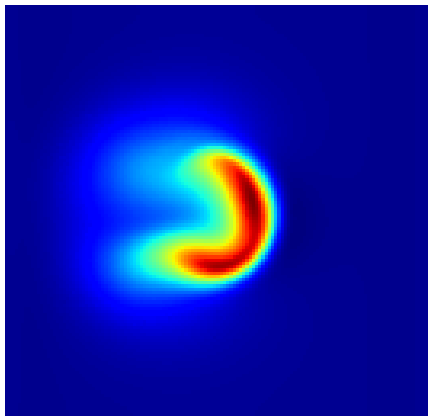
N.Walkden (CCFE)

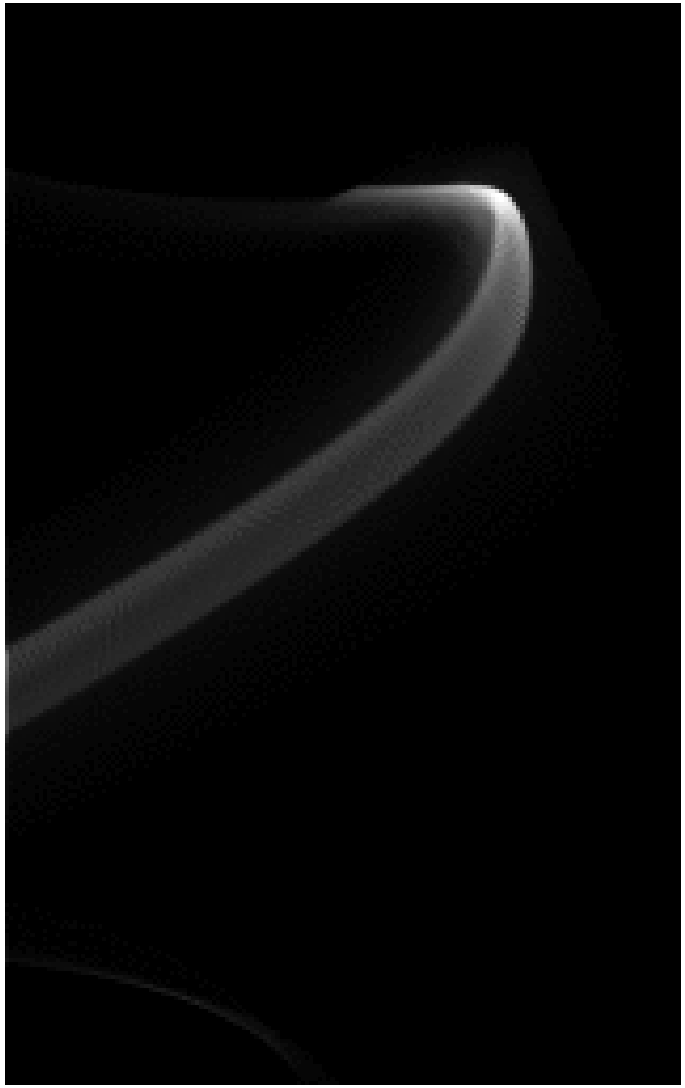
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





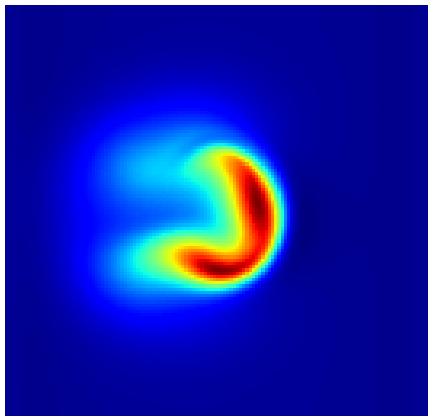
N.Walkden (CCFE)

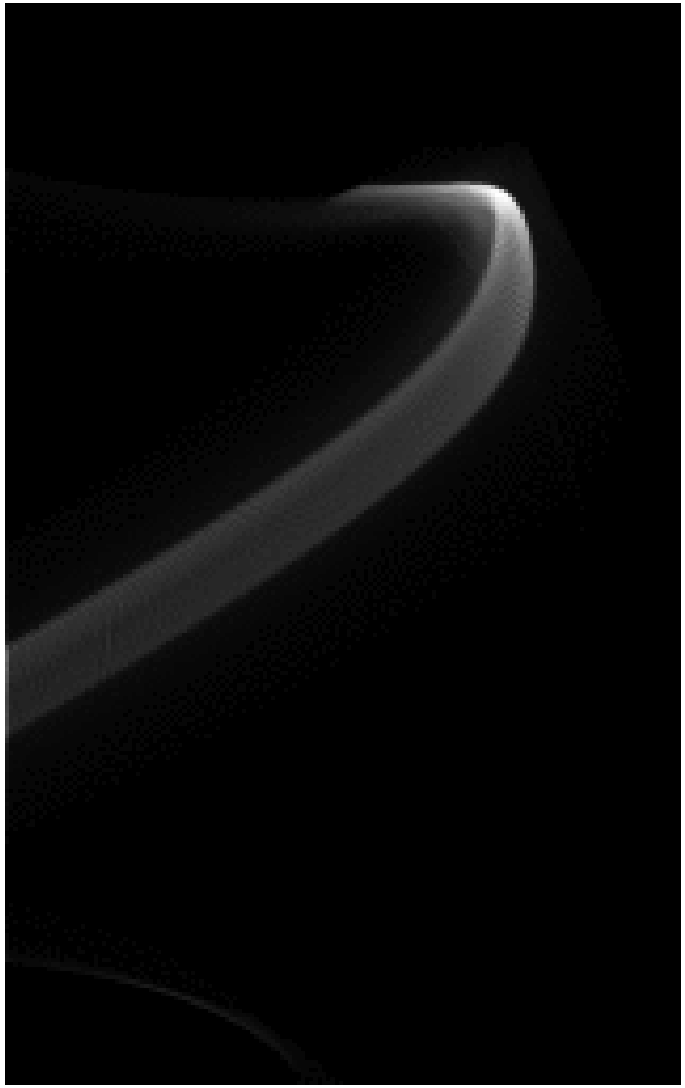
Development driven by physics needs

A major focus in the UK is edge and divertor physics

Edge turbulence and blob dynamics

- Detailed measurements of blob characteristics
- BOUT++ simulations and synthetic diagnostics





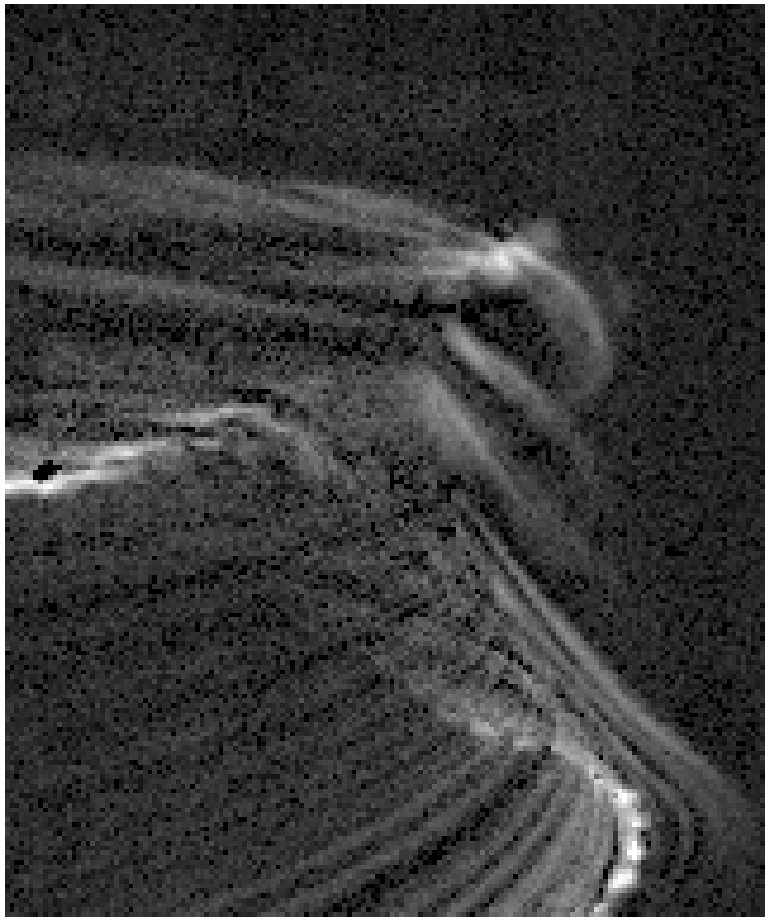
N.Walkden (CCFE)

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

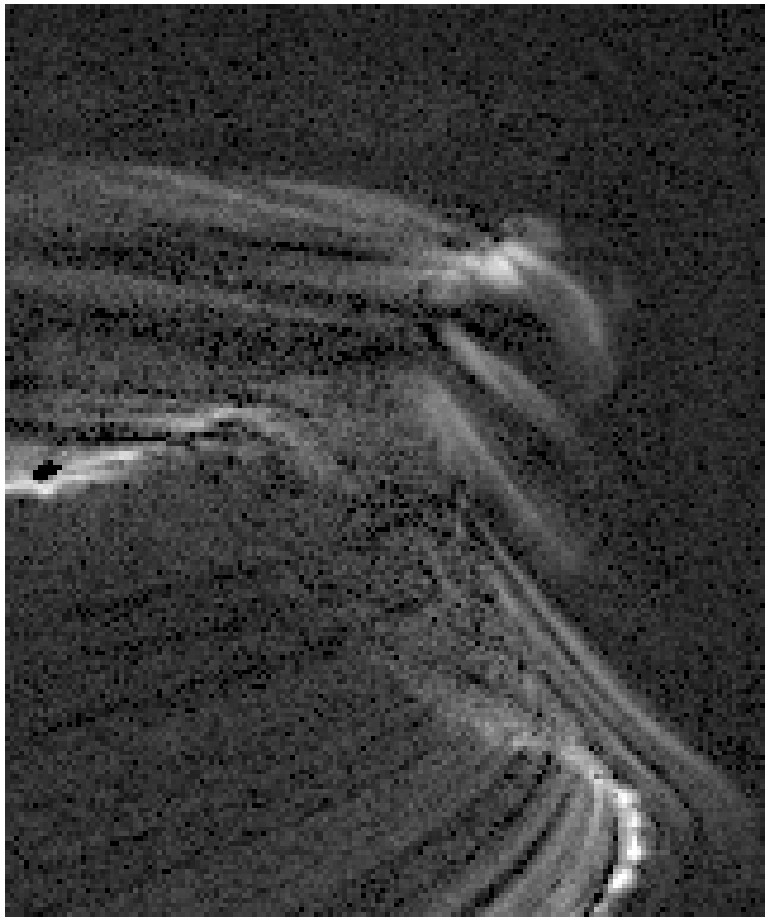
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

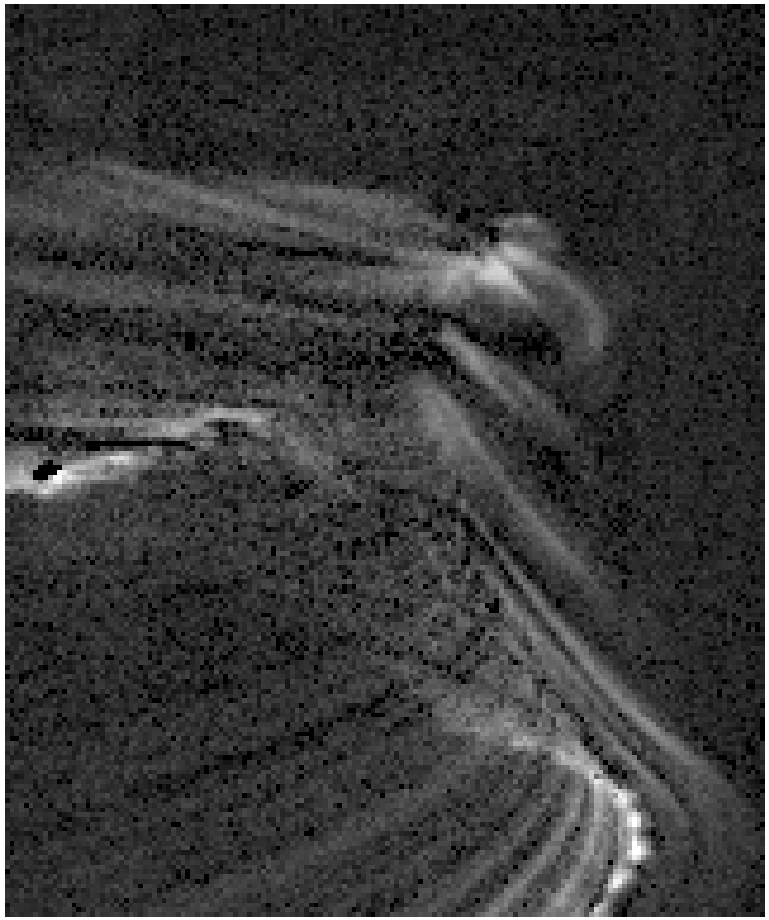
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

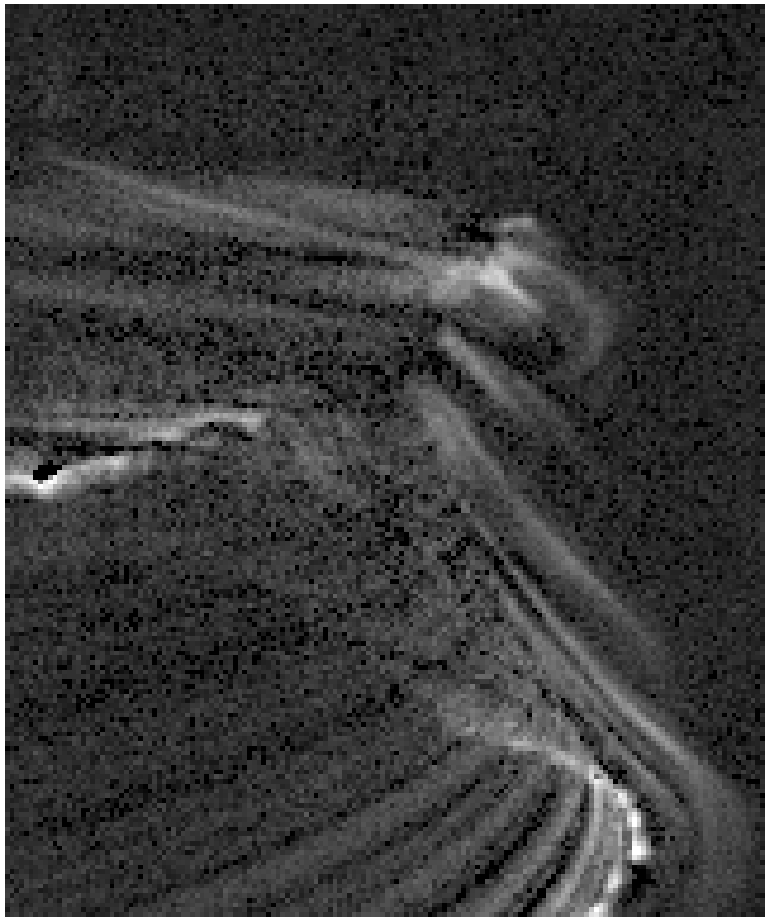
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

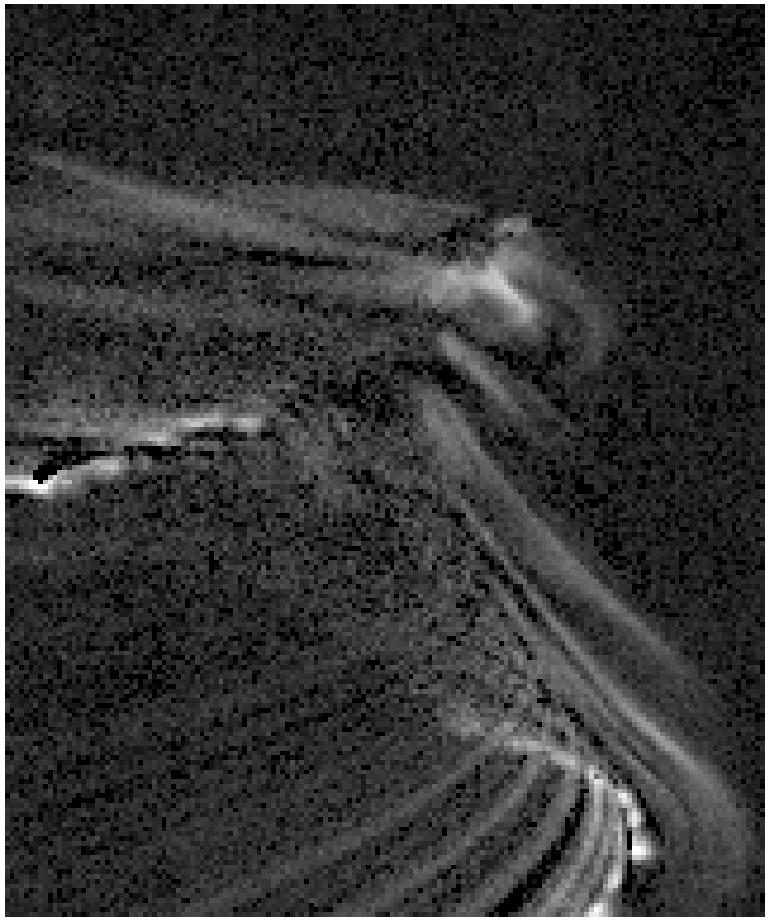
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

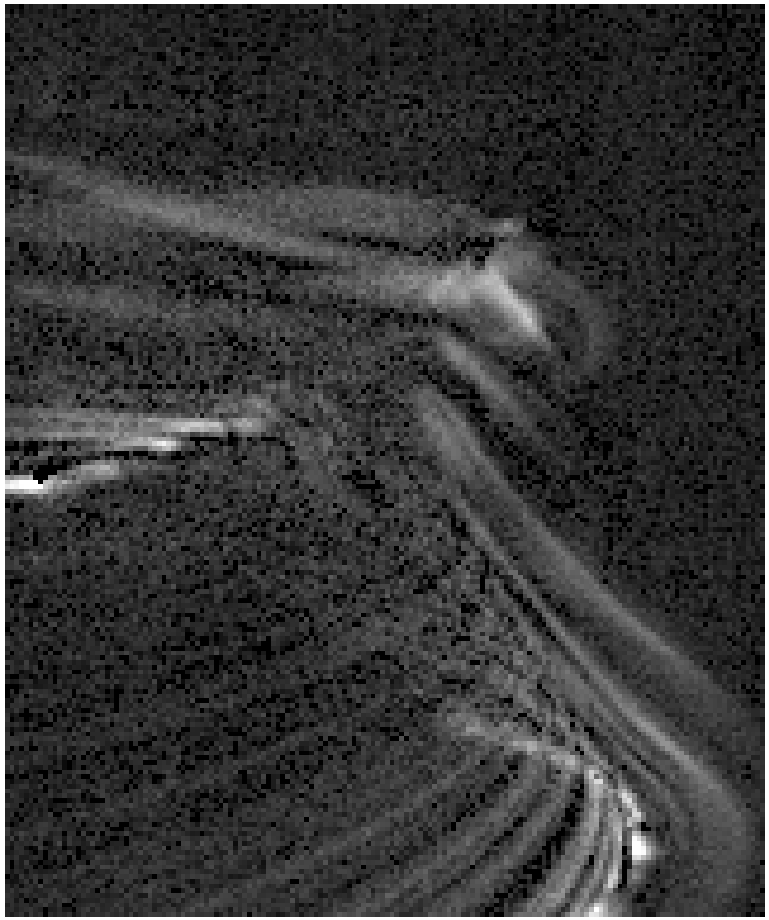
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

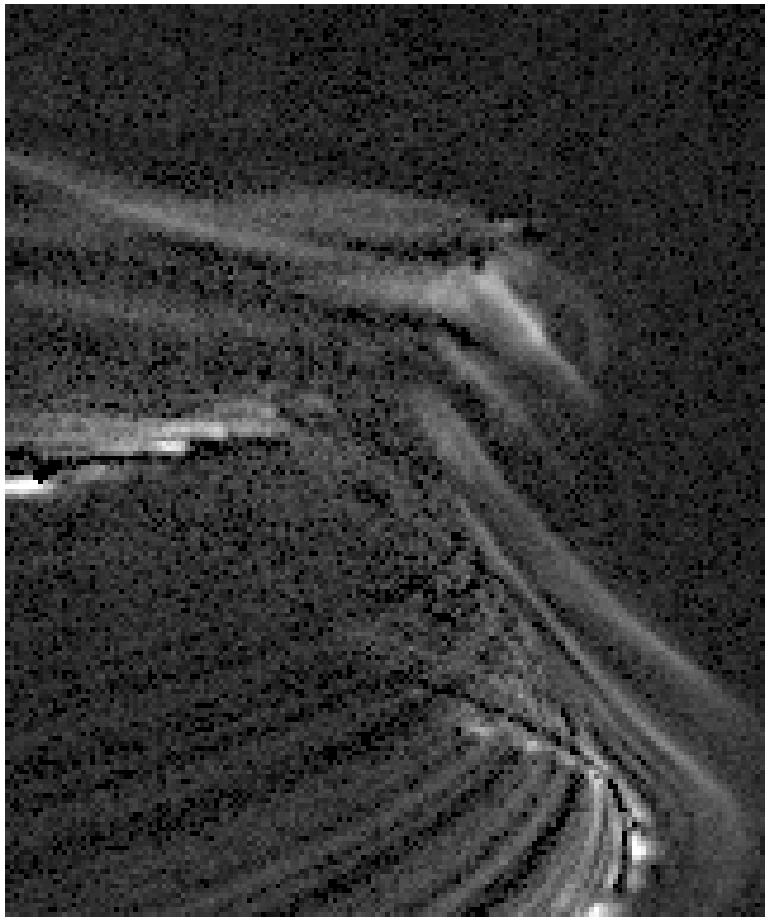
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

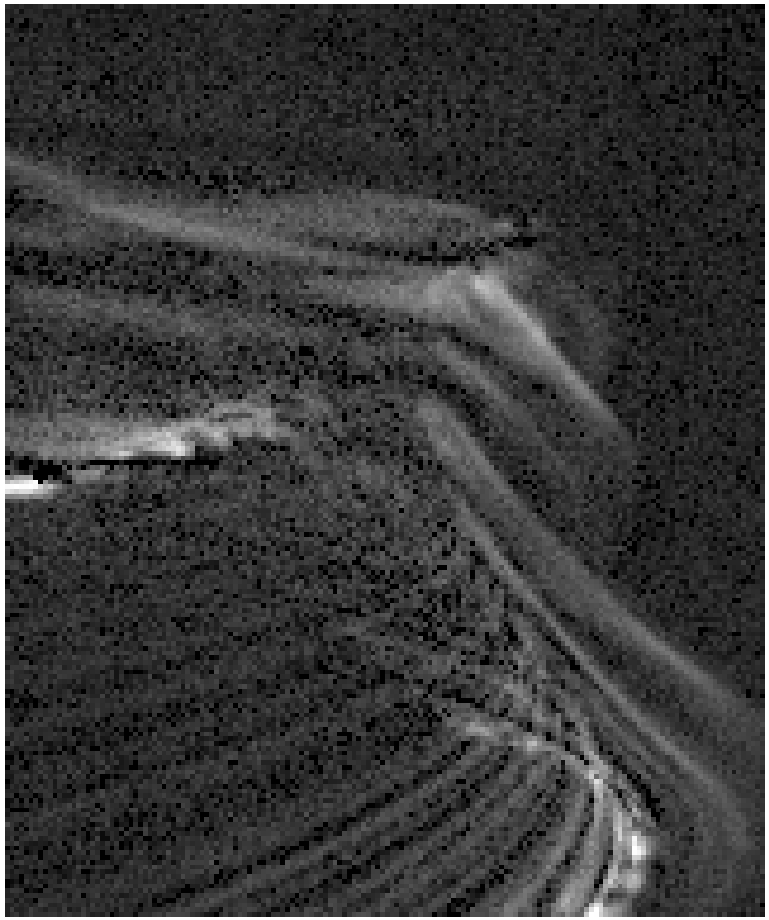
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

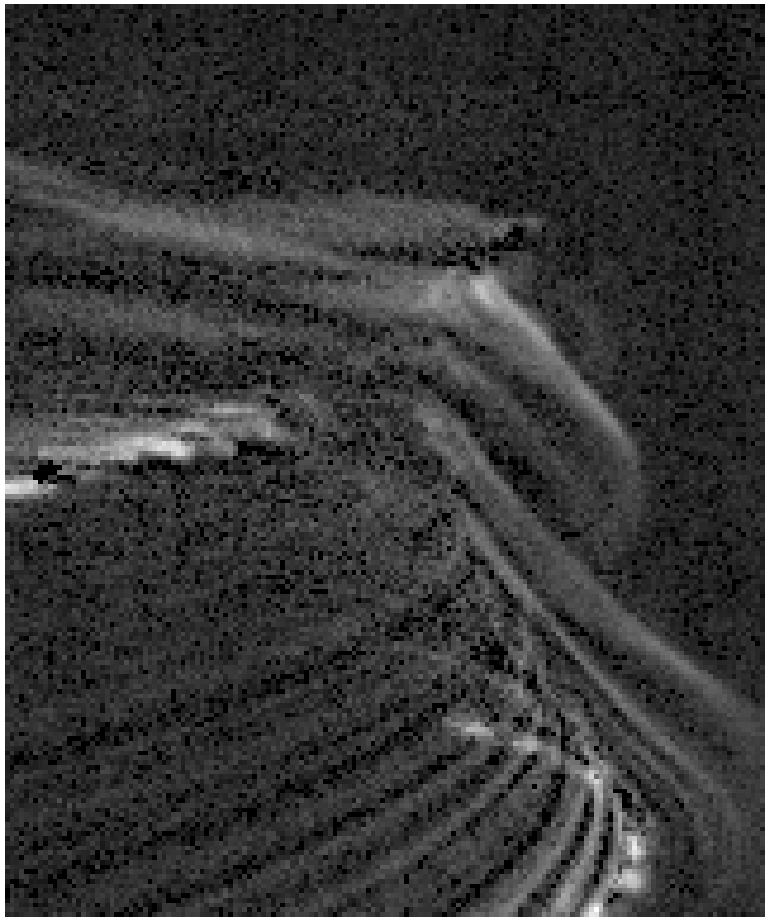
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

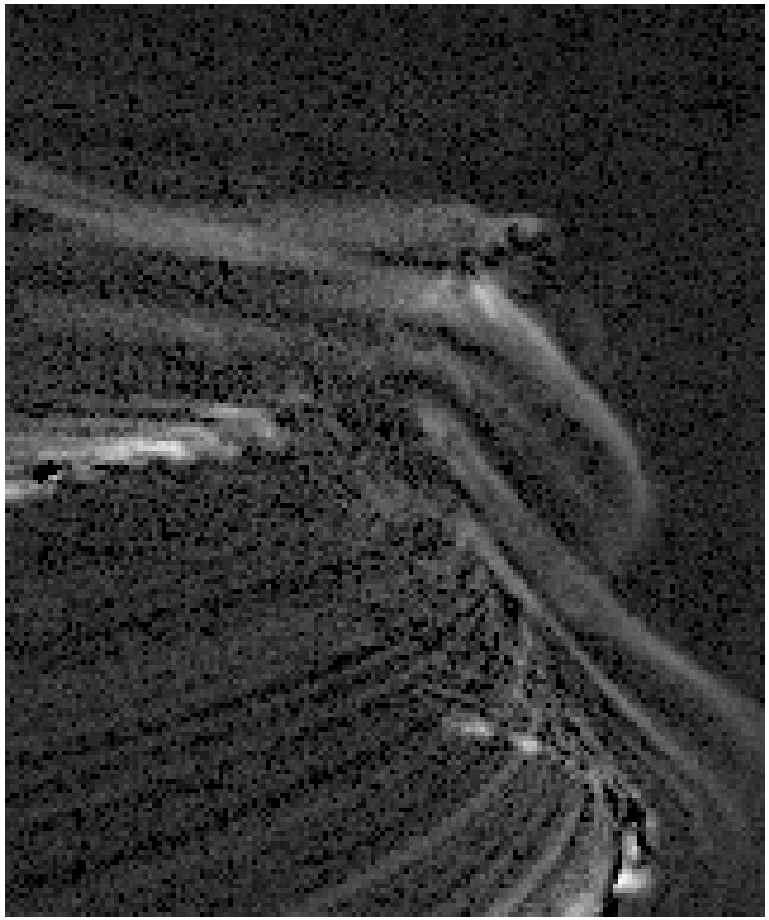
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important



Enhanced imaging of MAST divertor leg

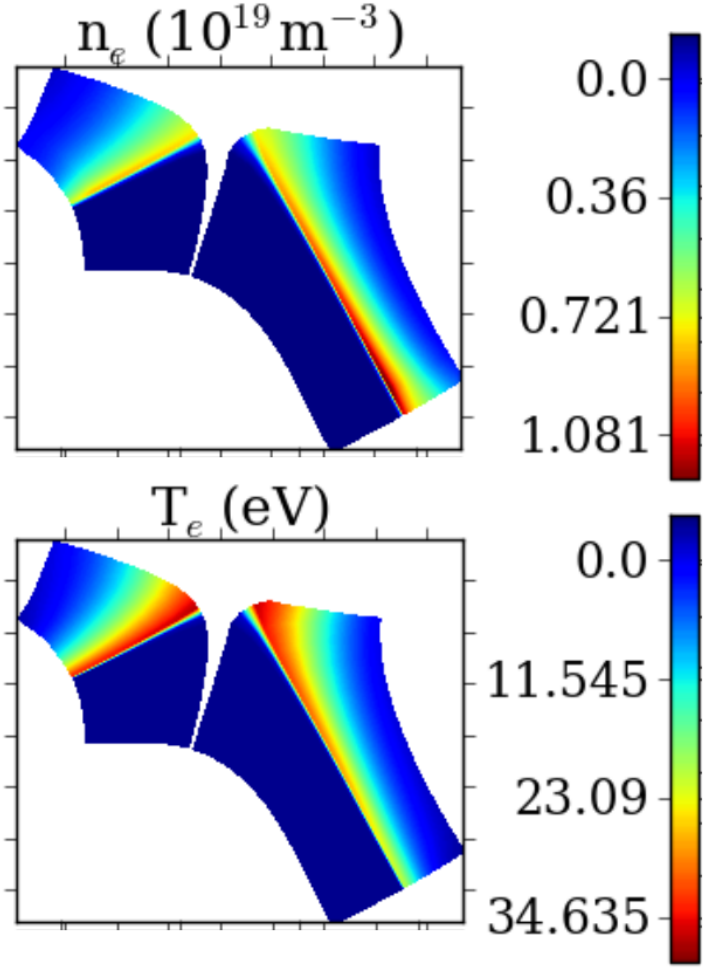
J.Harrison, CCFE

Development driven by physics needs

A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important
- Transport simulations with recycling



N.Walkden : Collisional electron transport in MAST.

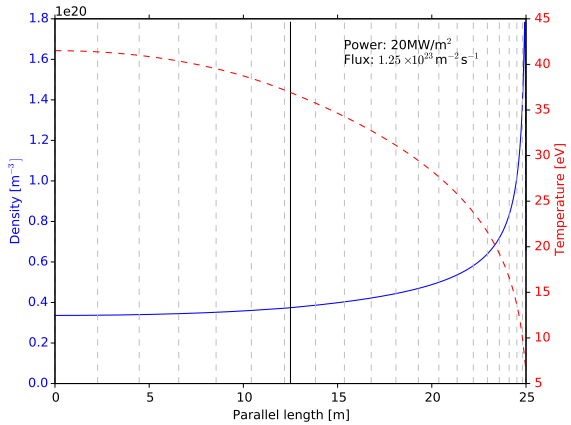
Submitted to PPCF (2015)

Development driven by physics needs

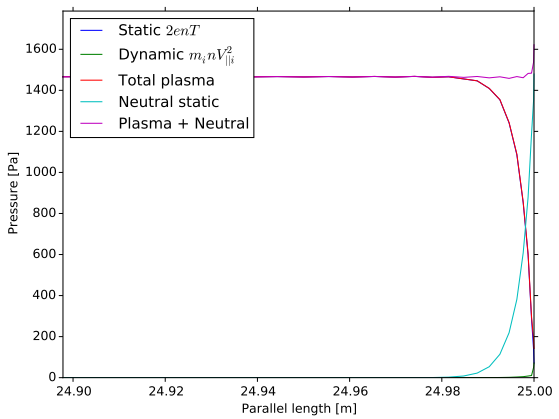
A major focus in the UK is edge and divertor physics

Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important
- Transport simulations with recycling
- High recycling and detachment



B.Dudson: 1D profiles in high recycling regime



Static, dynamic and neutral pressure

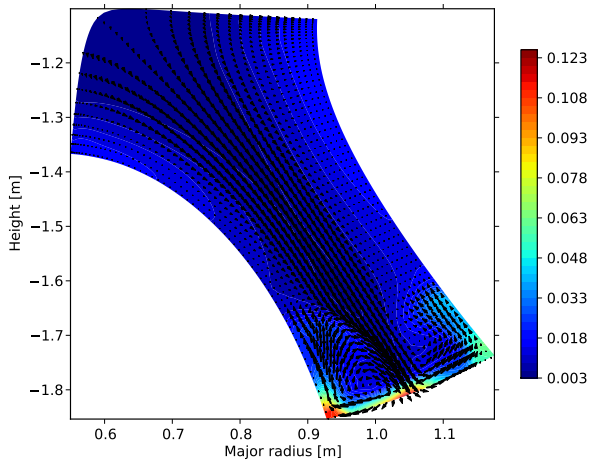
Development driven by physics needs

A major focus in the UK is edge and divertor physics

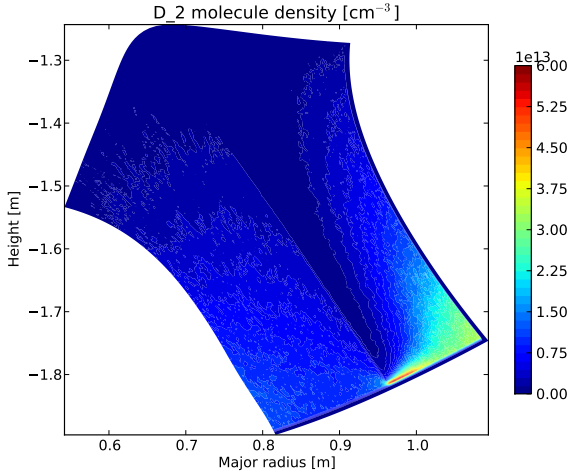
Divertor simulations and neutral gas

- Turbulence observed, role in heat flux spreading unclear
- Interaction with neutrals and impurities important

- Transport simulations with recycling
- High recycling and detachment
- Neutral gas, fluid and kinetic (EIRENE)



B.Dudson, J.Leddy: Neutral fluid (Navier-Stokes)



S.Mekkaoui, B.Dudson: EIRENE simulations

Recent developments (since 2013 workshop)

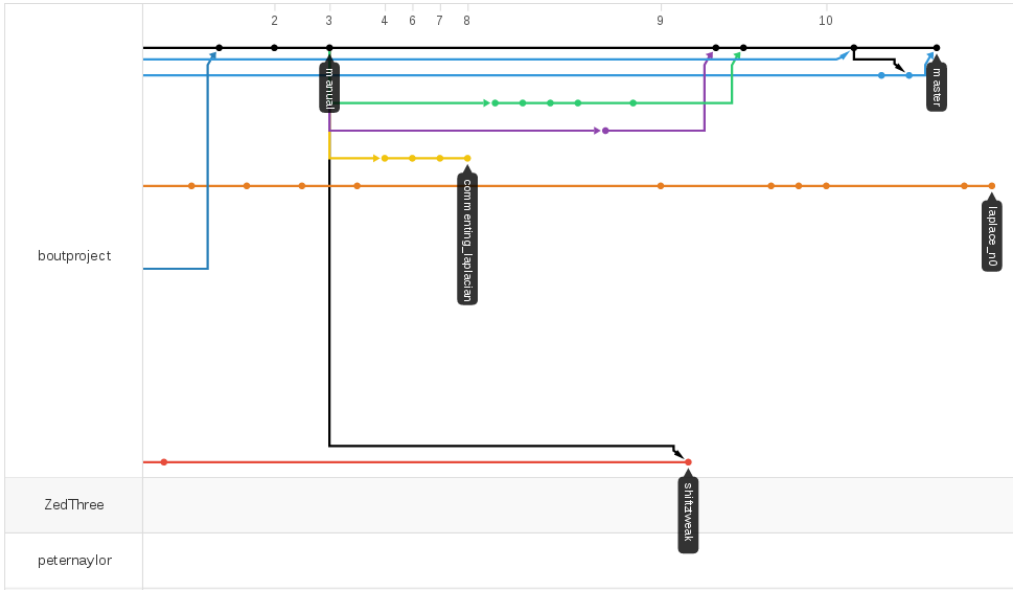
Many people contributing fixes, features and improvements :

George Breyiannis, Evan Davis, David Dickinson, Ben Dudson, Luke Easy, Erik Grinaker, Joe Henderson, Peter Hill, Jarrod Leddy, Michael Loiten, Jens Madsen, Peter Naylor, John Omotani, Kevin Savage, David Schwoerer, Luke Townley, Nick Walkden, Tianyang Xia

(apologies for omissions)

(`git diff --stat`)

- 688 files changed
- 76375 insertions(+)
- 26270 deletions(-)



Recent developments available now

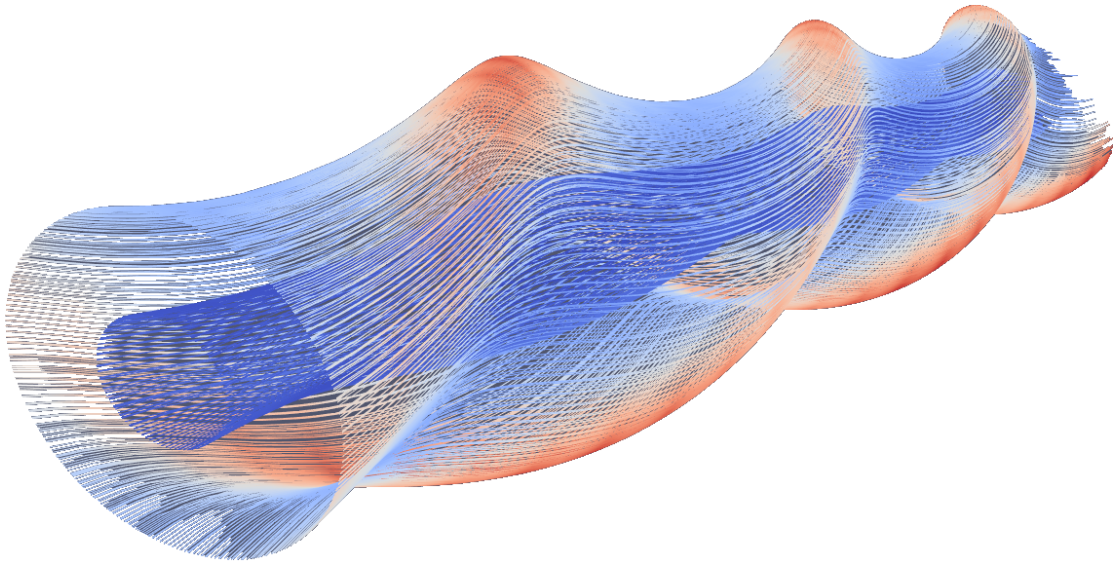
Some of the new features since September 2013:

- Verification with the Method of Manufactured Solutions (MMS)
- Analytic input expressions for mesh and external sources
- Improved, more consistent boundary conditions
- Implicit-Explicit (IMEX) schemes for time integration
- HDF5 file format support
- Python 3 compatible
- Explicit schemes: RK3-SSP, multi-order adaptive Runge-Kutta
- Many bug fixes, more documentation, new features...

Work in progress...

Some new features under development and still experimental:

- Finding eigenvector/eigenvalues using SLEPc (D.Dickinson, B.Dudson)
- Non-orthogonal meshes for realistic tokamak geometries (J.Leddy, N.Walkden)
- Interaction with neutral gas (B.Dudson, S.Mekkaoui, J.Leddy)
- Implementation of the Flux Coordinate Independent (FCI) scheme (P.Hill, B.Dudson, B.Shanahan)



P.Hill, B.Dudson, B.Shanahan: Straight stellarator

Verification

- **Verification**: Checks that a chosen set of partial differential equations is solved correctly and consistently
  - As the spatial and temporal mesh is refined the solution converges to a solution of the continuum equations
  - Order of convergence should be the accuracy expected of the numerical scheme used

- **Validation**: Checks that the correct set of equations has been chosen
  - Usually involves comparison against experiment

Verification methods

- Inspection of the output for “reasonable” features
- Comparison against analytic solutions
- Convergence to an analytic solution
- Code cross-comparisons
- Convergence to a manufactured solution

Manufactured solutions

How do you test convergence to an exact solution, when no analytical solution can be found for your equations?
→ Change your equations!

- If you are solving a set of equations to solve, for quantities $\underline{f}$:

$$\frac{\partial \underline{f}}{\partial t} = F\left(\underline{f}\right)$$

- Add a source term S:

$$\frac{\partial \underline{f}}{\partial t} = F\left(\underline{f}\right) + S\left(t\right)$$

- Now choose a (manufactured) solution $\underline{f}^M$ and calculate S

$$S = \frac{\partial \underline{f}^M}{\partial t} - F\left(\underline{f}^M\right)$$

- S can be calculated analytically, and evaluated to machine precision
- When the modified equations are solved numerically, any error must come from the discretisation of F or time integration.

Manufactured solutions: Example

By inserting a known source into the equations, a solution can be chosen for an arbitrarily complex set of equations [1]

e.g. Viscid Burger's equation:

$$\frac{\partial u}{\partial t} = \underbrace{-u\frac{\partial u}{\partial x} + \nu\frac{\partial^2 u}{\partial x^2}}_{F} \underbrace{+S}_{\text{Added source}}$$

Choose a solution for u

$$u = \sin\left(x - t\right)$$

Insert it into the equation to calculate the source S

$$S = \underbrace{-\cos\left(x - t\right)}_{\partial u/\partial t} + \underbrace{\sin\left(x - t\right)\cos\left(x - t\right)}_{u\cdot\partial u/\partial x} + \underbrace{\nu\cdot\sin\left(x - t\right)}_{-\nu\partial^2 u/\partial x^2}$$

[1] With some restrictions

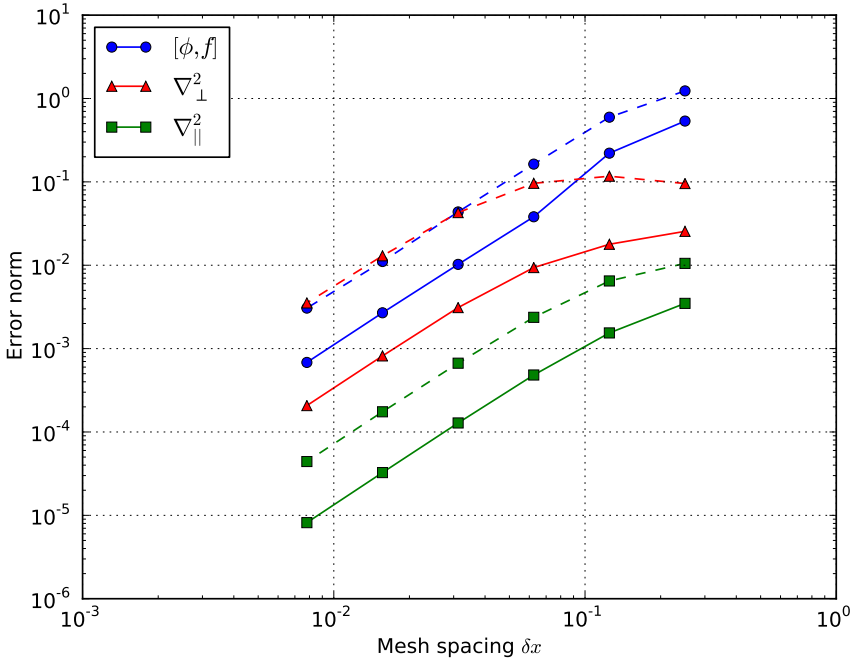
Manufactured solutions: Example

- The simulation code is now modified slightly, adding a time-dependent source to the equations
- Start at $t = 0$ with the manufactured solution
- Run the simulation for a short time Δt
- The difference between the exact and numerical solution $\epsilon = f(\Delta t) - f^M(\Delta t)$ at $t = \Delta t$ is due to numerical error
- This error should decrease towards machine precision as the resolution (time and spatial) of the simulation is increased (i.e. converge).
- The rate of convergence at high resolution ("asymptotic" regime) should agree with the expected rate e.g. $\epsilon \propto \delta x^2$ for second-order central differencing.

Verification using the Method of Manufactured Solutions

The Method of Manufactured Solutions (MMS) provides a rigorous way to check that the code is solving the equations correctly

- MMS has been successfully used for large parts of the code
- Testing largely automated
- Used routinely as part of test suite
- Analytic solutions and source functions in input file




```
[n]

solution = 0.9*xl + 0.2*sin(5.0*xl^2 - 2*zl)*cos(10*t) + 0.9

source   = 0.9*x - 1.0*(0.5*x - sin(3.0*x^2 - 3*z)*cos(7*t))*sin(pi*x)
         - 1.0*(-20.0*x^2*sin(5.0*x^2 - 2*z) + 2.0*cos(5.0*x^2 - 2*z))
         *cos(10*t) + 0.4*(pi*(0.5*x - sin(3.0*x^2 - 3*z)*cos(7*t))
         *cos(pi*x) + (-6.0*x*cos(7*t)*cos(3.0*x^2 - 3*z) + 0.5)*sin(pi*x))
         ...
```

Analytic inputs

Any value in the input options or mesh can be an analytic function

- External sources easy to modify
- Many meshes can be specified analytically: slab, cylinder, simple tokamak
- No mesh generation step needed
- All inputs in one place

Analytic inputs

Any value in the input options or mesh can be an analytic function

- External sources easy to modify
- Many meshes can be specified analytically: slab, cylinder, simple tokamak
- No mesh generation step needed
- All inputs in one place

Cylindrical coordinates **`examples/lapd-drift/pisces`**

```
[mesh]
...
Rmin = 5e-3     # minimum radius in meters
Rmax = 2.5e-2   # maximum radius
Rxy = Rmin + (Rmax - Rmin) * x
dr = (Rmax - Rmin) / (nx - 4)

dx = Bpxy * Rxy * dr
dy = length / ny
```

Time integration

The default time integration scheme is PVODE/CVODE, part of the SUNDIALS suite

- Fully implicit scheme (BDF with JFNK)
  - No CFL stability limit on timestep
  - Timestep instead limited by accuracy
  - Automatically adjusts order and timestep to optimise speed
  - Frequently does a good job with little input from user
- Some problems require preconditioning to be efficient
- Explicit methods can in some cases be quicker
- IMEX schemes treat parts implicitly, parts explicitly

Implicit-Explicit (IMEX) methods: example

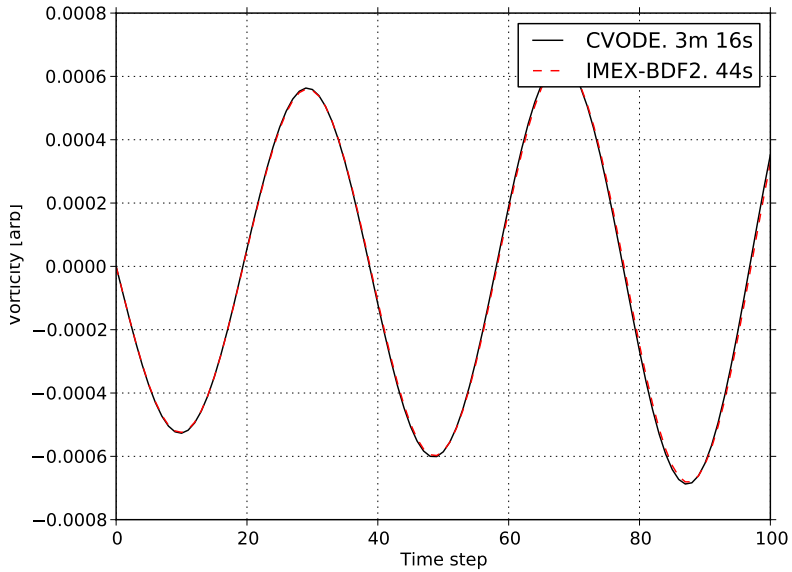
Several examples under `examples/IMEX`

Resistive drift wave

$$\frac{\partial n_e}{\partial t} = -[\phi, n_e] - \nabla \cdot (\mathbf{b} V_{\|e})$$
$$\frac{\partial U}{\partial t} = -[\phi, U] - \nabla \cdot (\mathbf{b} V_{\|e})$$
$$U = \nabla_\perp^2 \phi$$
$$\nu V_{\|e} = \mathbf{b} \cdot \nabla \phi - \mathbf{b} \cdot \nabla n_e$$

which is split into explicit and implicit parts.

Solved using IMEX-BDF2[1]



More on timesteppers Friday morning...

[1] W.Hundsdorfer, S.J.Ruuth JCP 225 (2007) 2016-2042

Experimental features: Eigenvalue solver (D.Dickinson)

BOUT++ is a nonlinear initial value code, so why develop an eigenvalue capability?

- A lot of physics understanding comes from the linear behaviour of a system: mode frequencies and growth rates.
- For many studies, BOUT++ runs performed are linear simulations, where the aim is to find the growth rate. This requires long simulations in order to identify an eigenmode, and can only find the fastest growing mode.
- Allows more detailed comparison / benchmarking against linear codes such as ELITE or GATO
- Allows exploration of linear modes in a range of fluid and gyro-fluid models

Eigenvalue solver (D.Dickinson, B.Dudson)

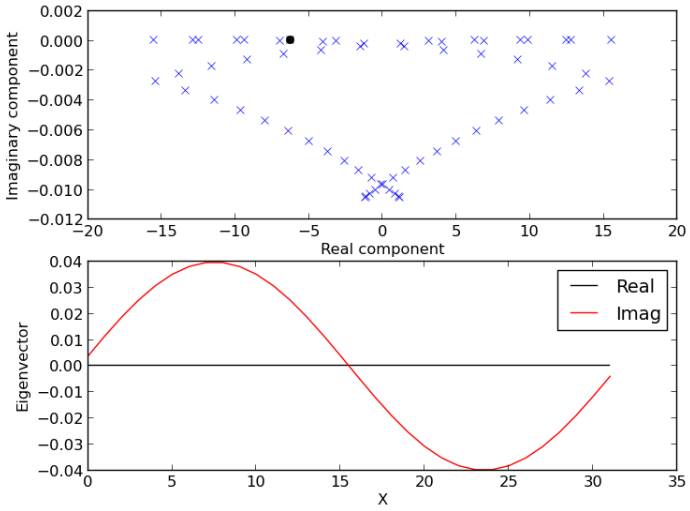
- Want use of eigensolver to be easy for the user and developer
  →Implement as a type of time solver.
- Dont write our own eigenvalue library: **Use the SLEPc library**
  Builds on top of PETSc (which BOUT++ already supports).
- Once BOUT++ has been compiled with SLEPc support user can just select solver:type=slepc in any of their linear physics models.
- In practice some knowledge of what eigensolver is doing is useful!

Eigenvalue solver example: eigen-box

Wave equation in a 1D box with fixed boundaries

$$\frac{\partial^2 g}{\partial t^2} = \frac{\partial^2 f}{\partial x^2}$$

- Clearly identifies physical mode spectrum with zero growth rate

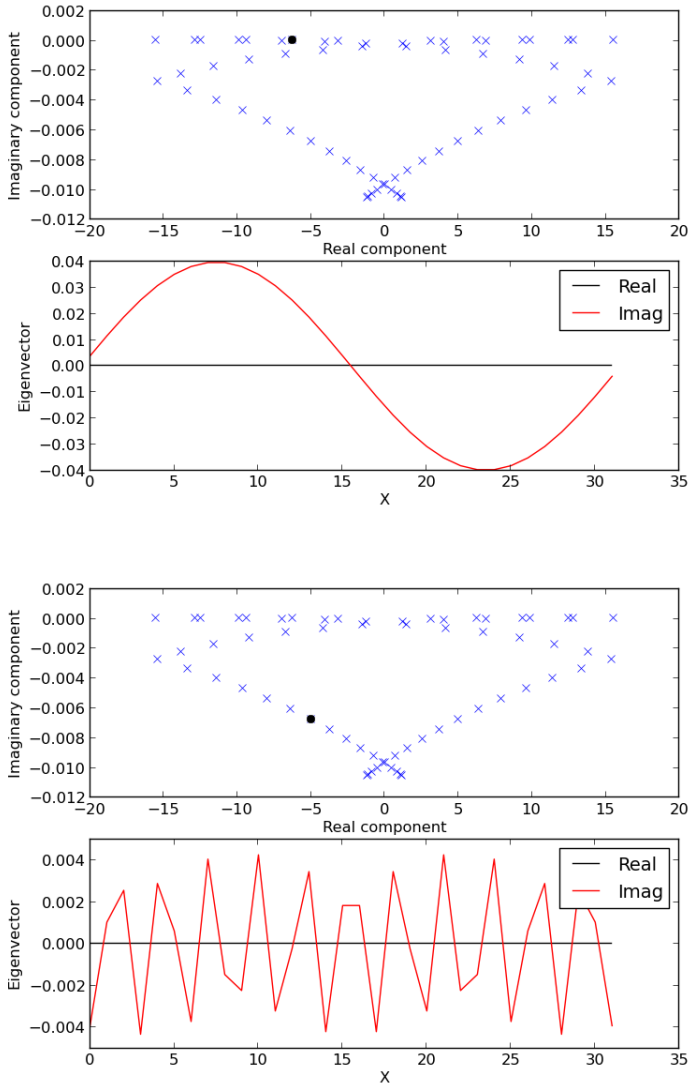


Eigenvalue solver example: eigen-box

Wave equation in a 1D box with fixed boundaries

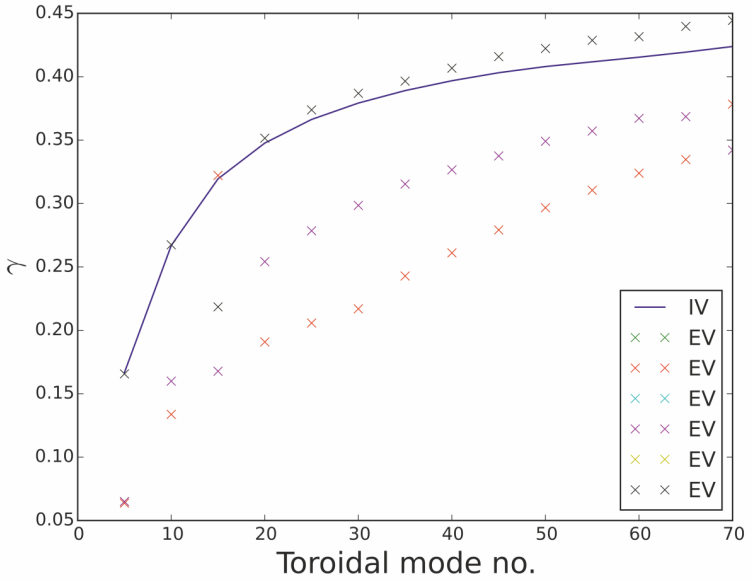
$$\frac{\partial^2 g}{\partial t^2} = \frac{\partial^2 f}{\partial x^2}$$

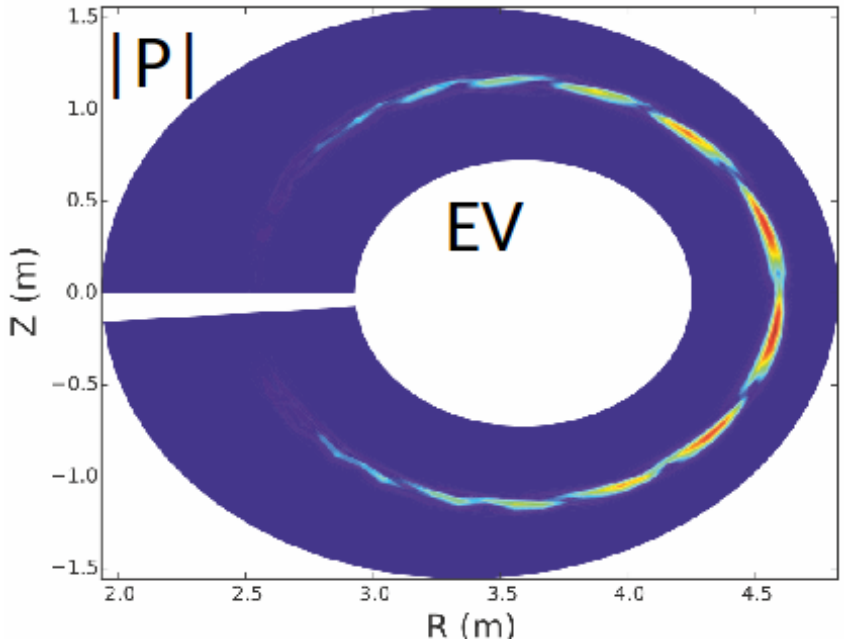
- Clearly identifies physical mode spectrum with zero growth rate
- Numerical (unresolved) modes also found, with negative growth rates
- Example includes python script to plot spectrum. Click on eigenvalue to show eigenvector

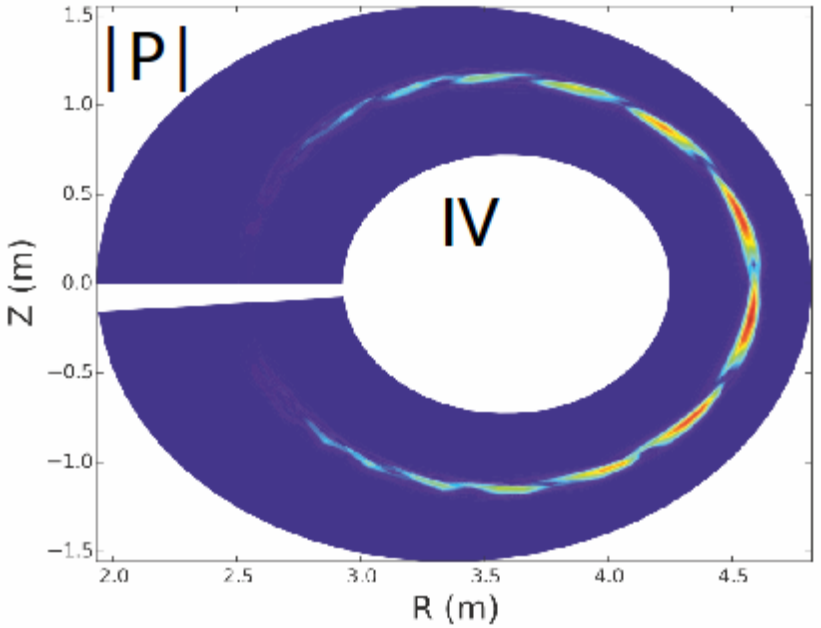


Eigenvalue solver example: elm-pb (D.Dickinson)

- Comparison of growth rates from linear initial value runs (green line), and SLEPc eigenvalues (crosses)
- Mode structure matches
- Sub-dominant modes also found, under investigation







Conclusions

- BOUT++ is a collection of useful algorithms and routines for solving quite general PDEs in 1,2 or 3-D
- Functionality mainly driven by tokamak edge physics needs
- Very active development, growing community

Purpose of this workshop

- To discuss key physics and computational issues related to the edge of fusion devices
- To share new developments, and prepare researchers to use and further develop BOUT++
- To promote collaborations within the BOUT++ community and beyond